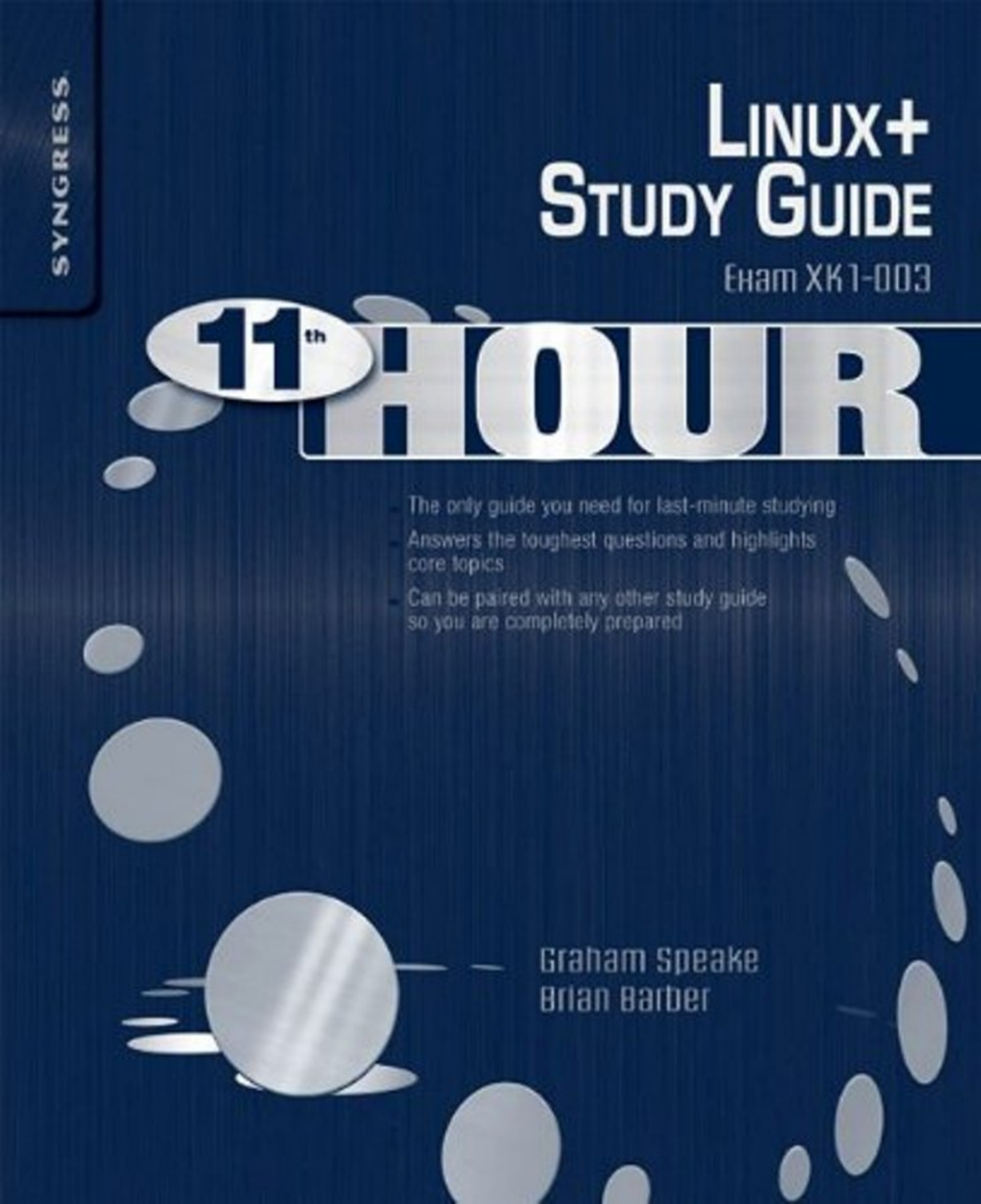
SYNGRESS
LINUX+
STUDY GUIDE
EXAM XK1-003
11th
HOUR
The only guide you need for last-minute studying
Answers the toughest questions and highlights core topics
Can be paired with any other study guide so you are completely prepared
Graham Speake
Brian Barber

Syngress is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
Linacre House, Jordan Hill, Oxford OX2 8DP, UK

Eleventh Hour Linux+: Exam XK0-003 Study Guide

Notices
Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data
Application submitted

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN: 978-1-59749-497-7

Printed in the United States of America
09 10 11 12 13 10 9 8 7 6 5 4 3 2 1

Typeset by: diacriTech, Chennai, India

Author Bios

LEAD AUTHOR

Graham Speake (CISSP #56073, M.Inst. ISP) is a risk management consultant with BP, one of the world's largest energy companies. He currently provides risk assessment and remediation consultancy to BP operating units throughout the world. His specialties include industrial automation and process control security, penetration testing, network security, and network design. Graham is a frequent speaker at security conferences and often presents security training to BP staff around the world. Graham's background includes positions as a consultant at ATOS/Origin and an engineer at the Ford Motor Company.

Graham holds a bachelor's degree from the Swansea University in Wales and is a member of the ISA. Graham was born in the United Kingdom, but now lives in Houston, Texas, with his wife, Lorraine and daughter, Dani.

CONTRIBUTING AUTHORS

Brian Barber (Linux+, MCSE, MCSA, MCP+I, MCNE, CNE, CNA-GW) works for the Canada Deposit Insurance Corporation (CDIC) as a project manager and architect for CDIC's IT service management program. He first started using Linux at home with Red Hat 5.1 and since then he has been a staunch advocate of open source software, belonging to the Ottawa Canada Linux User Group (OCLUG) since 2001 and the Ottawa Python Authors Group. His primary areas of interest are operating systems, infrastructure design, multiplatform integration, directory services, and enterprise messaging. In the past, he has held the positions of Principal Consultant with Sierra Systems Group Inc., Senior Technical Coordinator with the LGS Group Inc. (now a part of IBM Global Services), and Senior Technical Analyst at MetLife Canada.

He has been coauthor, technical editor, or lead author for over 15 books and certification guides. He is an experienced instructor and courseware developer. Recently, he was a Contributing Technical Editor for *Cisco Router and Switch Forensics: Investigating and Analyzing Malicious Network Activity*, (ISBN: 978-1-59749-418-2, Syngress), and *Cisco CCNA/CCENT Exam 640-802, 640-822, 640-816 Preparation Kit*, (ISBN: 978-1-59749-306-2, Syngress).

Terrence V. Lillard (Linux+, CISSP) is an IT Security architect and an expert in cybercrime and cyberforensics. He is actively involved in computer, intrusion, network, and steganography cybercrime and cyberforensics cases, including investigations, security audits, and assessments both nationally and internationally. Terrence has testified in U.S. District Court as a Computer Forensics/Security Expert Witness. He has designed and implemented security architectures

for various government, military, and multinational corporations. Terrence's background includes positions as principal consultant at Microsoft, the IT Security Operations manager for the District of Columbia's government IT Security Team, and instructor at the Defense Cyber Crime Center's (DC3) Computer Investigation Training Academy program. He has taught IT security and cybercrime/cyberforensics at the undergraduate and graduate level.

He holds a B.S. in Electrical Engineering, Master of Business Administration (MBA), and is currently pursuing a Ph.D. in Information Security.

Chris Happel has over 20 years of experience with voice and data networking and security. He is currently a managing consultant for Liberty Trio, LLC, and is an avid supporter of GNU/Linux and open source software.

TECHNICAL EDITOR

Kevin Riggins (CISSP, CCNA) is a Senior Information Security Analyst with over 20 years of experience in information technology and over 10 years of experience in information security. Kevin has used and managed Linux systems since 1995. Kevin has technical and strategic experience in a broad range of technologies and systems. Kevin currently leads the Security Review and Consulting team at Principal Financial Group which performs information security risk assessments and provides information security consulting services for all business units of The Principal. He holds a B.A. in Computer Science from Simpson College, Indianola, IA, is a member of ISSA, Infragard, and is the author of the Infosec Ramblings blog.

CHAPTER 1

Installing Linux

Exam objectives in this chapter

- A Note about Hardware
- Installing from Local Media
- Installing across the Network
- Laying out the Filesystem
- Disk Types

INTRODUCTION

Initial operating system installation requires the understanding of important concepts including computer hardware, environment settings, partitions and network settings. Successful installations need good planning performed in advance.

A NOTE ABOUT HARDWARE

Linux is supported on many different hardware platforms because of its flexible architecture and the support of the open source community.

Fast Facts

The Linux hardware compatibility architecture is divided into four categories as follows:

- Central processing unit (CPU) architectures supported continue to expand and include 32- and 64-bit Intel and AMD architectures, SPARC.
- Hardware abstraction layer (HAL) is designed to function as a tier between the physical hardware and software functioning on the system. Its purpose is to hide hardware complexity and differences from the operating system *kernel*.
- The monolithic kernel architecture is designed to function dynamically and supports the loading of modules and instructions to implement all

operating system services. The Linux kernel integrates the CPU architecture via a series of device drivers and kernel extensions.

- Hardware components are presented as device drivers, and for each *Linux distribution* a unique Linux Hardware Compatibility List is created.

For the Linux+ exam, you will need a fundamental understanding of the following components:

- Power supply units (PSU) are the devices required to provide the various computer hardware components within your system with direct current (DC).
- Motherboards are circuit boards housed inside the system unit that contains the CPU, the memory (RAM) chips, and the slots available for expansion cards. Other expansion boards interface with it to receive power and to provide bidirectional communications.
- CPUs are responsible for data processing and control the functions performed by the various hardware components, process all software instructions issued, and determine the speed of the system.
- Memory stores the data or programs that the CPU processes.
- Expansion boards are devices that expand the capabilities of a computer by enabling a range of different devices.
- Video adapters are expansion cards that translate binary into the images viewed on the computer monitor.
- Storage devices are internal and external devices used for storing data and information.

Crunch Time

Although the Linux+ exam is not about the hardware, you need to have a basic understanding of the components to install and support Linux and be able to answer the following:

- CPU make and architecture (32 or 64 bit)?
- How much RAM (memory) do you have?
- What installation media do you have (DVD/CD/network)?

INSTALLING FROM LOCAL MEDIA

The purpose of this section is to present you with the major Linux installation decisions required during the operating system installation process.

Linux Installation Process

Linux can be installed from various local and network sources. Apart from a few unique settings, the installation process for each installation source is essentially the same. Linux local media installation can be performed by using CD, DVD, or .iso sources. When the installer is launched from a local source, the system completes an on-board self test and displays an initial screen allowing various options, such as install, repair, rescue, and so forth. The options offered will vary depending upon the Linux distribution, but typically include the following:

- **Boot from the Hard Disk** It is used to automatically boot an existing Linux operating system.
- **Repair Installed System** This option repairs a previously installed system.
- **Installation** This option loads the mini Linux kernel from the Linux distribution and starts the installation process.
- **Rescue System** This option starts a specialized small Linux kernel without a graphical user interface. It loads the Linux Kernel into RAM and can modify configuration files, check the filesystem for defects, verify and/or modify the boot loader configuration, resize the partition, and a few other critical system modifications that may be necessary.
- **Check Installation Media** It is used to ensure the image's integrity.
- **Memory Test** This option conducts systematic tests of your system RAM using memtest86.

The installation process is divided into three distinct stages: *preparation*, *installation*, and *configuration*. The preparation stage assists you in configuring the system's language, date and time, desktop environment, user account information, user and root password authentication methods, and type of disk partitioning information. The installation stage is a noninteractive process, which installs the software and prepares your system for the initial boot sequence. The final stage is the configuration stage and, depending on whether you selected automatic or manual configuration, the installer software will either preconfigure various network settings or allow you to input network configuration information. For example, your machine's host name, domain name, and other network configuration settings (for example, *interfaces, DHCP, firewalls*).

Preparation

The first stage, preparation, collects information from you regarding your system's environment and your preferences. The default language and keyboard settings are English (US). Selecting the language and keyboard settings will automatically switch the system to the prescribe settings. The Installer application performs a system analysis of your system by conducting a system probe to search for various storage devices (for example, USB, Firewire, floppy disks, hard disk drives), existing Linux partitions and system files, determining whether the system can be updated, and launching the Package Manager.

The Clock and Time Zone setting allows you to set the Region, Time Zone, and system Date and Time information. In addition, you can determine if you would like to use local time or UTC (Coordinated Universal Time). The system can be configured to use Network Time Protocol (NTP) after the installation process is completed. Some distributions allow you to chose which desktop environment you wish to install (if any), with GNOME and KDE being the most popular.

PARTITIONING

Partitioning is the process of selecting and implementing a partition and file-system schema based upon your disk layout. The choice to select the default partition-based option or the logical volume management (LVM)-based option will normally be given. In addition, the user can choose to edit the existing partitions or create new partitions. The decision to create a new partition or edit an existing partition depends on if the system will coexist with an existing operating system (for example, Microsoft Windows) or contains more than one disk drive, or if you want to resize a foreign filesystem partition (for example, New Technology File System (NTFS)).

Crunch Time

Correctly partitioning a hard disk to install and support Linux is essential. The recommended partitioning schema represents the most common approach that often entails having two *primary partitions* and one *extended partition*. The two primary partitions support the *root* partition and the *swap* partition. The extended partition supports the *home* partition. A hard disk can have a maximum of four primary partitions.

USER SETTINGS

User accounts can be created along with local or network authentication. You may be able to assign the same password to the system administrator "root" account, but for better security, it is best to use different ones. Optionally, you may be able to set an Automatic Login feature to allow the system to automatically log you into the system whenever the system restarts (or reboots). For security reasons, this option should not be selected and users should always be required to enter their usernames and passwords.

Fast Facts

The user information will include the following:

- **User Full Name** User's first and surname must be entered.
- **Username** Username for logging in to the system.

- **Password** It can be alphanumeric, case sensitive, and should not contain any accented characters or umlauts. The system automatically check's for weak passwords.

Installation Settings

During installation, Linux is installed in accordance with the choices made previously. A number of Linux distributions, such as openSUSE, will display the choices you made prior to installation for you to confirm these and, optionally, modify them prior to installation occurring.

The last step in the installation stage prepares your system to boot into the new Linux operating system, which includes copying of the system files to the system, saving any system configurations, installing the Boot Manager, saving any installation settings, and finally preparing the system for the initial boot.

INSTALLING ACROSS THE NETWORK

Network source installation is slower than installing from local media due to limitation on network bandwidth.

Fast Facts

For network installation, you need to remember:

- Primary network protocols available are Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Network File System (NFS), and Server Message Block (a protocol) (SMB).
- Installation can occur from an Internet-based server using HTTP or FTP.
- The targeted Linux system must be started from a network bootable CD or DVD image.
- The IP-address or name of the server is required for HTTP, NFS, or FTP installations.
- Installation can use an anonymous FTP server.

The system will complete an on-board self test as before and load the Linux kernel that will provide you with various menu options for installing Linux and additional configuration options required prior to the installation process. While the desired network installation source is varied (for example, DVD, SLP, FTP, HTTP, SMB, Hard Disk) for the Linux+ Certification exam, we will concentrate on the HTTP, FTP, and NFS protocols.

Table 1.1 Network Installation Parameters

Network Installation Type	Domain Name/IP Address	Distribution Source Directory/Folder	User Name	Password
HTTP	Required	Required	N/A	N/A
FTP	Required	Required	Required (anonymous login if empty)	Required
NFS	Required	Required	N/A	N/A

Table 1.1 summarizes the type of network installations performed, and the parameter values you will need to provide during the initial installation.

After selecting the preferred network-based installation method, the installer will work as described earlier in the Installing from Local Media section.

Crunch Time

Network installations entail making a small Linux boot disk and downloading the majority of the code, which will take considerably longer than installing the system with local media. Although you can install Linux on a number of systems simultaneously, ensure you have adequate bandwidth.

LAYING OUT THE FILESYSTEM

There are two options on laying out the filesystem: *partition-based* (default) or *LVM-based*.

Fast Facts

Before deciding on how to layout your system, you need to consider the following:

- Physical hard disks have limitations imposed by the disk manufacturer (for example, disk geometry) and by the system manufacturer (for example, BIOS).
- Primary partitions (maximum four per disk) divide the hard disk drive into physical groups.
- Extended partitions can be subdivided into *logical partitions* (groups).

- For any bootable operating system, there must be at least one primary partition which will be used by the operating system to store the kernel and a few pertinent files.
- The remaining three partitions can all be primary partitions, extended partitions, or a combination of both.
- For a dual-boot system, there must be two primary partitions (one for each operating system).

Partitions can be created or edited during the installation process, and the decision to create a new or edit an existing partition should be based upon factors such as if the Linux system will coexist with an existing partition or contains more than one disk drive, or if you want to resize a foreign operating system's partition (for example, NTFS).

Another approach to view, edit, and create partitions is via the use of the *fdisk* command line tool. The *fdisk –l* command displays all disk drives attached to the system and their corresponding disk geometry. The output from the command shows the disk device name and size, the disk geometry, the disk identifier number, and the existing partition map (if a disk drive is accessible).

The *fdisk* command mode is used to create or modify partitions, entered by typing *fdisk* and the disk drive of interest. The *fdisk* command mode screen is displayed and the menu command *m* is used to display a help screen that lists commands available for use.

The partitions created, whether primary or extended (with logical subdivided partitions), must be assigned a partition type which are used for hosting specific filesystems. The partition type 83 is used to support Linux filesystems, and the partition type swap is used to support the swap partition (type 82). To select and configure the partition type, execute the *fdisk* command. Then type **t <enter>** and, when prompted, the partition number that you wish to change.

User, application, and system files and folders must be stored on the physical disk and a filesystem structure must be implemented on the physical disk to achieve this. The Linux environment provides support to many different filesystems, (for example, *ext2*, *ext3*, *ReiserFS*, *JFS*, *VFAT/NTFS*) each with its own unique way of storing files and folders internally for quick access and indexing. *Journaling* is a feature implemented in some filesystems to provide a mechanism to temporarily store information in a log (journal). The changes are stored in a log prior to the changes being implemented within the filesystem that reduces the amount of time required by a system recovering from a crash if the data was not updated to the filesystem.

There are three very common types of filesystems:

- **ext2** The Second Extended Filesystem (ext2), one of the oldest and most popular Linux filesystems, is the industry standard. It is a very reliable filesystem. The lack of journaling support is ext2 greatest weakness.

- **ext3** The Third Extended Filesystem (ext3) expanded the ext2 filesystem. ext3 provides journaling support and is the default filesystem on many newer versions of Linux.
- **ReiserFS** It performs faster than ext3 or ext2 and supports a larger maximum file structure (8TB).

Another approach to assigning filesystems to partitions is via the use of the *mkfs* command line tool. This command line tool assigns the filesystems (for example, ext2, ext3, and ReiserFS) to partitions. To assign the filesystem using the *mkfs* command, append the –t option followed by the desired filesystem type. For example:

```
mkfs -t ext /dev/sda
```

The assigning of partitions and filesystems to your hard drive may result in the filesystem or partitioning being corrupted or you may need to reclaim disk space from an empty and unmounted partition. To accomplish this task, you can use the *parted* command line tool, which can be used in reclaiming disk space, but it will also copy a filesystem from one partition to the next. During the partition creation process, *parted* will create the filesystem.

To use this tool, boot the system and select **Rescue System** from the options presented. The system will allow you to log in as root (no password is required). To enter the *parted* command mode, type ***parted*** and the disk drive of interest. For example: *parted /dev/sda*.

DISK TYPES

There are two additional disk types that need to be discussed: logical volume manager (LVM) and Redundant Array of Independent Disks (RAID). LVM is used to create logical volumes and RAID is used to improve performance and/or fault tolerance of the disk subsystem. The LVM-based and partition-based implementations can be configured on top of RAID or non-RAID systems.

LVM

LVM-based installations offer a unique approach for creating virtual partitions (also known as logical volumes). The standard partition-based approach, after implementation, is hard to change. The LVM approach offers greater control of the disk drive environment because you can create virtual partitions that can group physical partitions or disk drives together as one. The command-line tools for LVM are as follows:

- *pvcreate* prepares physical volumes for use in LVM.
- *vgcreate* creates and names volume groups.
- *lvcreate* creates and names logical volumes used by filesystems.

RAID

RAID uses your disk subsystem to provide enhanced read/write performance, protection against data lost due to disk failures, or both. It can be implemented

using hardware specific RAID controllers (Hardware RAID) or functionality embedded within the operating system (Software RAID).

DID YOU KNOW?

RAID subsystems require two or more disks to form one virtual disk. The differences between software and hardware RAID solutions are as follows:

- Hardware-based RAID performs faster than the software-based RAID implementation because software-based RAID requires more CPU time and has additional memory requirements.
- Software-based RAID is operating system dependent and hardware-based RAID is vendor independent.
- Hardware-based RAID may be more expensive than software-based RAID as it may require additional hardware components (for example, RAID controllers), unless incorporated on your motherboard.

RAID technology requires an understanding of three basic concepts: striping, mirroring, and parity. Striping joins the hard disk drives together to form one large disk drive. For example, three 300 MB drives joined together in a striping array will form a single 900 MB drive. Striping evenly writes data across all of the disks contained in the array and will evenly read data from all disks contained in the array, increasing overall disk subsystem performance. The downside to striping is that it does not provide any fault tolerance.

Mirroring forms one disk drive whose size is determined by the size of the smallest drive, and the same data is written to both drives, providing a level of redundancy in the event one drive crashes. The disadvantage of mirroring is the impact of having to record the same data twice across two different drives which reduces the disk subsystem performance. Parity stores information in the disk array subsystem that can be used to rebuild files or lost data in the event one of the disks in the disk subsystem array fails. Unlike striping and mirroring, parity requires a minimum of three disks inside the disk array subsystem. Each of these techniques is assigned a different RAID level.

RAID Levels

Each RAID level offers advantages and disadvantages. In general, the partitions should be stored on different drives to get the performance and fault tolerance you want. RAID levels can be concatenated (also known as nested) which means that the common RAID level numbers are combined with other RAID levels, sometimes with a "+" in between. For example, RAID 10 (or RAID 1+0) consists of a RAID level 1 disk array subsystem, each of which is one of the "drives" of a level 0 disk array subsystem.

Fast Facts

The decision on what RAID level to use depends on whether you are using hardware or software RAID, the number of disks, and the resilience you require.

- Level 0 stripes data across two or more hard disk drives, with no fault tolerance.
- Level 1 mirrors data across hard disk drives with fault tolerance.
- Level 5 stripes data across three or more disk drives with fault tolerance.
- Software RAID supports RAID levels 0, 1, and 5.
- Level 0+1 is a striped array set in a mirrored disk array subsystem.

SUMMARY OF EXAM OBJECTIVES

In this chapter, we discussed how Linux was installed and some of the options that are available. The hardware architecture was described, including the processors Linux supports, the HAL, how the Linux kernel interacts with the hardware, the hardware components that are supported, and the Hardware Compatibility List. Installation of Linux from local media, as well as remotely across a network, was presented, highlighting the differences between the two approaches. The generic decisions to be made in installing the operating system were given to ensure that any Linux distribution could be used. The Linux filesystem was introduced and the various types of filesystems that could be created were discussed. The primary and extended partition types for hard disk partitioning were explained and when to use each. In particular, at least one primary partition is needed to hold the bootable operating system. Finally, LVM and RAID were described to show how to create logical volumes and how to stripe, mirror, and parity check disk arrays.

TOP FIVE TOUGHEST QUESTIONS

1. When using the *mkfs* command, what is the "-t" option used for when inserted as a parameter?
   - **A.** The "-t" option is used to test the network bandwidth.
   - **B.** The "-t" option is used to terminate the operating system.
   - **C.** The "-t" option is used to assign filesystems to partitions.
   - **D.** There is no "-t" parameter associated with the *mkfs* command.
2. To see all the current disk drives on your system and the current disk geometry, what command should you enter?
   - **A.** *mkfs –t*
   - **B.** *flpart –l*
   - **C.** *fdisk –l*
   - **D.** *diskgeo –t*

3. What is the purpose of the *parted* command?
   - **A.** To reclaim unused disk space.
   - **B.** To establish disk striping.
   - **C.** To implement RAID 5.
   - **D.** To test system's on-board memory for defects.

4. You are installing Linux on your organization's server. This is a new installation. You must partition the hard disk for the new Linux installation. Which is the best hard disk partition architecture for supporting root, swap, and home partitions?
   - **A.** Primary partition architectures should be used for the root and swap partitions and the home partition should use the extended partition architecture.
   - **B.** The root, swap, and home partitions should all be extended partitions.
   - **C.** The root and home partitions should be placed on extended partition architectures and the swap partition should be placed on the primary partition.
   - **D.** Only swap and home should be placed on the primary partition and the root partition should not be used.

5. During the initial Linux installation process, which application is used to test your system's RAM for an x86-based CPU architecture?
   - **A.** testmemx86
   - **B.** memtest86
   - **C.** memtestx86
   - **D.** memx86test

ANSWERS

1. Correct answer and explanation: **C**. Answer **C** is correct. The parameter to assign a filesystem using the *mkfs* command is "-t." For example, *mkfs -t ext3 /dev/sda*

   Incorrect answers and explanations: **A**, **B**, and **D**. Answers **A**, **B**, and **D** are incorrect; all three are false statements.

2. Correct answer and explanation: **C**. Answer **C** is correct. The *fdisk –l* command is used to view the current disk drives and see the current disk geometry.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect; *mkfs –t* is the command used to assign a filesystem to a disk partition. Answers **B** and **D**, both are fictitious Linux commands.

3. Correct answer and explanation: **A**. Answer **A** is correct. The *parted* command is used to reclaim unused disk space on a disk partition.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answers **B** and **C** are performed after the space was been reclaimed. **D** is a system procedure that is performed by using the memtest86 application.

4. Correct answer and explanation: **A**. Answer **A** is correct. The primary partition should be imposed in the root and swap partitions to place constraints and boundaries around the partition. The home partition should be extended to allow for growth as a user's home directory grows or more users are added to the system. In addition, for an operating system to boot, one of the partitions must be a primary partition to support and store the system's boot software and operating system files.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answers **B**, **C**, and **D** are incorrect; all three are false statements because the system will never be bootable. In each case, the root partition is placed on an extended partition. This approach would prevent the system from booting. The placing of the swap partition on an extended partition is permissible, but it is not a good system design.

5. Correct answer and explanation: **B**. Answer **B** is correct. memtest86 is a standalone memory test application for x86-based systems.

   Incorrect answers and explanations: **A**, **C**, and **D**. Answers **A**, **C**, and **D** are incorrect; all three are fictitious application names.

CHAPTER 2

Managing Filesystems

Exam objectives in this chapter

- Filesystem Types
- Mounting and U(n)mounting Filesystems
- Partitions
- Directories
- Filesystem Management

INTRODUCTION

This chapter presents the strategies for the creation of filesystems, the different types of filesystems, the tools used to create filesystems, and the tools used to administer filesystems.

FILESYSTEM TYPES

A filesystem provides the operating system with a framework for the storage, organization, modification, removal, and retrieval of digital information. To achieve these requirements, filesystems are responsible for organizing and maintaining files, folders, metadata, and residual data as containers for storing digital information. Regardless of the selected filesystem type, data containers can be as large as terabytes (TB) in size or as small as a sector. The size of a sector can vary among the different filesystem types; however, it is the smallest amount of disk space that can be assigned to hold a file. The default size of a sector is typically 512 bytes.

Fast Facts

Filesystems organize and maintain the following data:

- Files are sectors of allocated space within a filesystem used for storing digital information. The allocated space can be contiguous or noncontiguous. There are four different types of files: regular, links, first-in first-out, and sockets.
- Directories (or folders) are sectors of allocated space within a filesystem used to group files. This association is performed and maintained within a

File Allocation Table (FAT). Directories can be hierarchical structures, with directories containing subdirectories.

- Metadata are data used by the operating system to further characterize files and folders. Typically, some form of indexing file is created and used. The indexing information can contain the file size, the file date and time stamp, sector location, and information pertaining to access permission and device type.
- Residual data are a form of data remaining within the filesystem after the file, filename, folder, and folder name relationship has been severed and can vary in size. This typically occurs after deleting a file or folder and can be as small as a sector. Slack space is any residual data remaining on a disk not currently associated with any particular file.

For the various types of filesystems, the *storage containers* (also known as *storage media types*) can reside on many different storage devices. The storage devices can house static and dynamic generated data. As a result, different filesystem structures exist for the different types of containers. Not all operating systems provide support for the various storage containers.

DID YOU KNOW?

Listed below are descriptions of each of the storage media types supported by Linux:

- The hard disk storage media type is used for storing data either on internal or on external hard disk drives. This filesystem storage is known as *local filesystem storage*.
- The optical storage media type is divided into two different categories, supporting CDs and DVDs. Both CDs and DVDs are supported by the ISO 9660 filesystem standard, also known as the *CD File System* (CDFS). DVDs also support the Universal Disk Format (UDF), which is considered to be a CDFS replacement.
- The network-based storage media type is an architectural model comprising client and server computers, for example, Server Message Block (SMB) and Network File System (NFS). The client establishes remote network connectivity to a server to access files.
- The removable storage media type is a filesystem designed for storing files on flash memory devices. Flash memory devices are primarily used in memory cards and USB flash drives.
- The random access memory (RAM) disk storage media type functions as a storage container by using a segment of main memory in RAM. The Compressed ROM Filesystem (cramfs) is used with many Linux distributions for initrd images.
- Special-purpose storage media types are systems implemented to handle files, folders, and metadata dynamically. Special-purpose filesystems are created by an application or a software/system management tool.

Local Filesystems

Local filesystems alleviate many challenges that are typically associated with network-based filesystems, RAM disk-based filesystem, and removable filesystem. This includes performance challenges due to limit network bandwidth, access failure due to lost network connectivity, and nonpersistent filesystems (for example, RAM disk) that require reconfiguration once the electrical power is restored.

This section presents some additional filesystems that are somewhat common: FAT, New Technology File System (NTFS), and Virtual File Allocation Table (VFAT). FAT is a widely supported filesystem with a number of different standards including FAT12, FAT16, FAT32, and a variation called VFAT. These are summarized in Table 2.1.

EXAM WARNING

Unlike VFAT, earlier Windows operating systems that used the FAT filesystem allowed files to be named with only eight alphanumeric characters with a period separating the name from a three alphanumeric character extension. VFAT supports long filenames, which allows files to have names up to 255 alphanumeric characters. VFAT is the preferred filesystem for mounting foreign filesystems.

Network

Linux filesystems can also span across a network that allows them to function as a client/server model that provides file-sharing services to systems remotely. The server component provides shared directories that can be accessed via network connectivity. The client component, after obtaining access, connects the shared directories to a mount point on the local filesystem. Within the Linux environment, the two primary network-based filesystems are the NFS and the Server Message Block Filesystem (SMBFS).

NFS is a framework designed to allow a user on a client workstation to access remote files over a network on a network-based server. In this model, a NFS server shares one or more directories that can be accessed remotely by a network client. NFS uses registered ports TCP 2049 and UDP 2049.

Table 2.1 FAT Cluster and Partition Sizes

Name	Cluster Address Size	Partition Size
FAT12	12 bits	16 MB
FAT16	16 bits	2 GB
FAT32	32 bits	2 TB

SMBFS is a framework designed to allow workstations to access directory/file shares on a server. This model, implemented as client/server architecture, comprises two components. The first component, SMB protocol, provides mechanisms for performing interprocess communications and listens on port TCP 445. The second component, SMB service, is an application that resides on both the client-side and the server-side. The SMB service interoperates with the system's security authentication mechanisms and local filesystem, which allows the SMB environment to interoperate with various other SMB servers. The Microsoft version of the SMBFS is known as the *Common Internet File System* (CIFS). The Samba application, an open source client/server implementation, provides support for both SMBFS and CIFS.

Crunch Time

The difference between local and remote filesystems needs to be understood as well as the advantages and disadvantages of both.

- Common network (remote) filesystems are NFS and SMB.
- Connectivity with Microsoft systems is usually via SMB.
- Local filesystems can reside on internal or external attached devices.
- Local filesystems normally outperform network filesystems.

MOUNTING AND U(N)MOUNTING FILESYSTEMS

Mounting is a form of attaching or joining a separate storage device to the existing root directory hierarchy. It makes physically separate disks accessible and/or partitions on a local or remote machine available, with the attached location on the client machine called a *mount point*. Linux users can manually mount a filesystem or have a filesystem automatically mounted when the system initially starts up.

The *mount* and *umount* Commands

To manually mount a filesystem to your existing root directory structure, you can establish a connection to a local mount point using the *mount* command, which can be used with or without any arguments. The *mount* command issued with no arguments lists all the currently mounted filesystems on the system. A unique feature about Linux is its treatment of devices (for example, storage devices and terminals) as directories (folders) with the /dev directory used to mount devices. To mount a device, use the *mount* command with, typically, two arguments. The first argument represents the storage device that you wish to attach to your root directory structure. The second argument represents the directory location where you want to attach it underneath, the mount point. The general directory locations for most Linux systems are the /mnt and /media directories. The /media directory is typically used for the mounting of

removable media (for example, floppy disks, CD/DVD drives). In addition, prior to mounting any storage device, a mount point directory needs to exist before executing the *mount* command.

```
mount/dev/sdb1/mnt/morestorage
```

The *mount* command does include extra arguments (for example, *sb* for superblock and *noload* for turning off journaling) that can be used. To specify a specific filesystem type, the *-t* argument should be used. Normally, the Linux kernel typically will detect the type of filesystem of the storage device you are attaching.

To disconnect a device, use the *umount* command, specifying the mount point.

```
umount/mnt/morestorage
```

Fast Facts

Mounting and unmounting of partitions is an often used process by system administrators. Knowing the commands and options is very important.

- The command to unmount a drive is *umount* not *unmount*.
- Linux normally detects the type of filesystem you are attaching.
- Add filesystems to be permanently mounted to the /etc/fstab file.

/etc/fstab

The /etc/fstab file is text-based file, which lists the filesystems that are mounted on a permanent basis whenever the system is booted. The structure for adding a filesystem to the /etc/fstab file so that it can be mounted automatically at bootup contains the disk partition to be mounted, the directories mount point, the filesystem type, and other filesystem options (for example, *noload, noatime,* and *noauto*). A typical structure is shown in Figure 2.1.

```
LinuxExpert1@linux-01vc:/etc> cat fstab
/dev/sda1            swap                swap       defaults              0 0
/dev/sda2            /                   ext3       acl,user_xattr        1 1
/dev/sda3            /home               ext3       acl,user_xattr        1 2
proc                 /proc               proc       defaults              0 0
sysfs                /sys                sysfs      noauto                0 0
debugfs              /sys/kernel/debug   debugfs    noauto                0 0
usbfs                /proc/bus/usb       usbfs      noauto                0 0
devpts               /dev/pts            devpts     mode=0620,gid=5       0 0
LinuxExpert1@linux-01vc:/etc>
```

FIGURE 2.1
/etc/fstab file

PARTITIONS

Partitioning is the allocation of electronic storage space within a storage device into separate data areas for a specific filesystem type. Whether implemented either via a partition editor tool or via the *fdisk* command, partitions can be created, deleted, or modified. The folder and file creation and placement process cannot occur until after the partitioning and formatting of the storage device. The decision to implement one or more partitions per storage device is based on certain advantages and disadvantages.

Fast Facts

Creating multiple partitions on your storage device:

- Improves access time for data and applications colocated on the same partition
- Supports the separation of user files and operating system files
- Provides dedicated operating system swap or system paging file space
- Protects operating system files from the rapid growth of system data files (for example, logging and system cache) that may consume all available disk space quickly
- Provides support for multibooting environments
- Provides a layer of isolation to prevent or protect one partition's system resources from another partition's system resources
- Allows the implementation of various filesystems and different disk geometry strategies to improve read and/or write performance

After creating a partition, a filesystem must be assigned to it via a partition editor tool or via the *mkfs* command. A directory structure would then need to be created for the allocation and organization of files and folders. The balancing of the size and structure of the partitions, filesystems, and directories must be addressed, with decisions based on.

- Initial boot access: During the installation process, it is critical the system BIOS (basic input/output system) and GRUB (grand unified bootloader) functions are able to access a partition. The system BIOS will access the primary partition to execute the bootloader, GRUB, which must be able to access the /boot partition to retrieve the Linux kernel and other critical configuration files. As a result, for some installations, placing the initial boot applications and configuration files on a separate partition makes the installation process easier, especially as, during the initial boot process, the Linux kernel and initrd will have limited access to disk device drivers.

- Disk space growth: The design and implementation of a Linux system require disk space growth projections. The amount of disk space needed for current and future use needs to be calculated, and the projections should include installed applications and application data and user files. If a fixed partition-based implementation is used, space limitations or quotas should be used. The introduction of logical volume management (LVM) reduces disk growth limitations by allowing data in volumes to expand across separate physical disks.
- Security and permissions: Information stored on the same partition opens the door for various security risks. For example, if the partition is corrupted, all the data could be lost. In addition, directories not properly secured could give unauthorized users access to sensitive data (for example, everybody's HR records in a company). Creating separate partitions to segregate system resources from user files allows you to place more stringent access controls around system resources.
- Backup/restore: Creating several partitions provides greater backup flexibility, reduces backup performance impacts, and can improve restoration times.
- System repair/rescue: Creating a separate partition for critical partitions makes it easier to repair and/or rescue a corrupted partition.
- Logging/monitoring: Log files must be accessed, reviewed, and in most cases saved for security purposes. The creation of a separate partition for log files will make it easy to backup, restore, and secure these.
- Volatile/temporary data: Computers and associated applications are constantly creating volatile and temporary data. The Linux system automatically creates a swap partition to hold some of this data. However, there are other directories (for example, /tmp) that traditionally reside underneath the root directory. These other directories can also outgrow the disk space capacity of the partition.
- System maintenance: The creation of separate partitions for system administration and maintenance purposes can also make the job easier.

The decision to create a partition or not to create a partition is difficult. Clearly, a single partition system is not a sound system design; however, creating a partition for every directory is not feasible either. Table 2.2 provides more insight into why some directories are put on separate partitions.

The recommended partitioning schema is to have two primary partitions and one extended partition. The two primary partitions support the Linux *root* partition and the Linux *swap* partition. The extended partition supports the *home* partition.

EXAM WARNING

When deciding on the size of the partitions, bear in mind that the BIOS in some older computers may be able to access the first 1024 cylinders of a disk drive, approximately 528 MB.

Table 2.2 Linux Directories and Partitions

Linux Directory	Benefits of Separate Partition
/swap	The /swap directory is used as swap space and functions as virtual memory. This directory provides additional memory to your system to run large programs.
/boot	The /boot directory contains files and configuration information needed during the Linux boot process. The implementation of /boot under a different partition would make system rescues and repairs easier.
/home	The /home directory normally hosts users' account directories and user-specific files. This directory should be assigned a separate partition as the multiuser file may fill the partition. In addition, performing backup and restore functions is a lot easier if dynamically changing user files reside on a separate partition.
/tmp	The /tmp directory is used to support programs that require temporary file space that is volatile. It can reside under the root directory, but if temporary files grow quickly than your root directory, they may run out of disk space, perhaps from system or application core dumps.
/var	The /var directory contains system and application-generated information as a result of spooling, logging, and system temporary files. The /var directory should be placed on a separate partition, if possible. It is normally a subdirectory underneath the root directory. The primary reason why it should be separated is due to the possible growth of the log files and user mailboxes.

DIRECTORIES

The purpose of this section is to provide an overview of the key directories, as shown in Table 2.3. As a hierarchical tree structure, the filesystem starts at the top with a directory indicated by a forward slash called the *root directory* and contains all the underlying files and directories.

Fast Facts

The Linux filesystems can be considered as follows:

- A hierarchical structure to organize files and directories
- A directory structure based on the Filesystem Hierarchy Standard (FHS)
- Ordered consistently (due to FHS) to enable files and directories to be easily located by users and applications

Table 2.3 Critical Linux Directories

Linux Directory	Purpose
/	The root directory is the top tier directory and is the most important directory in the Linux system. It contains core directories and files, which include utilities, configuration files, bootloader information, and start-up information required for the initialization of the Linux system.
/bin	Contains Linux commands used by the system administrator and users, which are also accessible when the system is in single-user mode
/dev	Used within the Linux environment to mount devices
/etc	To control the execution of programs and support the dynamic Linux environment, the /etc directory is used. It contains system and application configuration files.
/media	Used to mount removable media (for example, floppy disks, CD/DVD drives, and USB/thumb drives) for access by the system administrator and users
/mnt	Similar to the /media directory, this is used to temporarily mount filesystems.
/proc	Functions as a virtual filesystem for system processes and the Linux kernel
/root	The /root directory, not to be confused with the root directory, is the home directory assigned to the *root* user account.
/sbin	Used to contain Linux utilities that are used only by the system administrator (for example, root). It contains executables for critical booting, restoring, recovering, and/or repairing the Linux system.
/usr/bin	Contains the primary executable commands on the system. Linux users and the root user can execute the commands in this directory.
/usr/lib	This contains software libraries and packages for programs, including object files and internal binaries that are platform specific.
/usr/lib64	Performs the same function as the /usr/lib directory but supports the 64-bit architecture
/usr/local	Location for locally installed applications. Software installed in this directory typically is not affected by system software updates. In addition, software installed in this directory can be shared.
/usr/share	Used to store read-only architecture neutral files. The files contained in this directory are platform independent.
/var/log	Contains data files generated as a result of spooling, logging, and system temporary files

When addressing all other second tier directories and files, using the full path naming convention, all subsequent files and folders use the forward slash to indicate their hierarchical position to prevent confusion with other directories that could have the same name but located at a different sublayer tiers (for example, third level and fourth level).

FILESYSTEM MANAGEMENT

There are tools to manage filesystems either locally or remotely, and these will be used by all system administrators, often on a daily or weekly basis.

Fast Facts

The filesystem tools that you will need to learn for the exam and will likely use on a regular basis are

- *du* provides a summary of disk space used per file in the current directory and disk space allocated for files contained in any subdirectories.
- *df* provides a summary of the amount of disk space available on a filesystem in many different ways by using different arguments.
- *df –i* displays information about inodes rather than file blocks.
- *df –h* presents a disk space summary in an easy-to-understand output format using kilobytes, megabytes, and/or gigabytes.
- Filesystems need to be maintained and should be checked and, if necessary, repaired using the *fsck* command.
- Filesystems need to be unmounted prior to running *fsck*.

Quotas

Users and errant applications, without limitations, can continue to add data to a folder that would eventually use up all the available disk space on a partition. Once a partition is full of data, not only will users and applications be unable to save information to disk but damage to system or application files may also occur. The implementation of disk quotas can provide valuable filesystem management support to allow you to specify limits on disk storage that may be allocated to a user or a group of users.

To implement disk quotas, you must add the following qualifiers "usrquota" or "grpquota" to each desired partition in the */etc/fstab* file and then reboot the system. Then, the *quotacheck* command needs to be executed to examine the quota-selected filesystems and build a table of the current disk usage for each filesystem with disk quota enabled. This information is stored in aquota.group and aquota.user files. Finally, the *edquota* command, a quota editor, is used to display and change the various quota settings for one or more users or groups.

After the disk quota system is implemented, a summarized disk quota review for a filesystem is available using the *repquota* command, which creates a summarized disk quota report. The report includes a summary of the disk quotas for the specified filesystems and summaries for the current number of files and amount of disk space per user. Finally, in case of a system crash and other filesystem failures, the *quotacheck* command is used to scan a filesystem for disk quota use and to create, check, and, if necessary, repair disk quota systems.

Loopback Device

Linux offers support for a loopback device, which supports the transformation of a special file containing an image of another filesystem into a device that can be used like any other Linux partition or device. Linux loopback devices are commonly used for CD/DVD ISO images. The disk image created of the CD/DVD disc contains the UDF or ISO 9660 filesystem format. Prior to accessing the loopback device, the ISO image must be downloaded and mounted. The Linux *mount* command is used to attach the virtual filesystem image.

NFS

NFS is a client/server model designed to make specified directories on a server available to a select subset of clients or all clients on a network. The server-side must configure directories for sharing, known as *exports*, and the client-side must be configured for mounting the exports. The shares (exports) made available via NFS are listed in the /etc/exports file. This file contains a listing of directories (exports) and the client machines that may mount the exports. Each line represents a shared directory and any associated options (for example, permissions).

The parameters for each line are as follows:

```
exported_directory <client1> (<options>) <clientN> (<options>)
```

- exported_directory is the directory being exported on the server.
- <client1> is the host or network to which the export is being shared for access. The client1 parameter can be based on Internet Protocol (IP) address, host name, or domain name for a single host; a wildcard (*) for a group of machines; or an IP network range.
- <options> are those imposed on the connection, which include read-only (ro), read-write (rw), root_squash (to prevent remote root access), no_root_squashing (to allow remote root access), and others.

To activate the access of shared directories (exports), the *exportfs* command can be used. The following set of arguments can be used with the command:

-a	To load and export all directories listed in the /etc/exports file
-r	Rereads the /etc/exports file after changes have been made to the share permissions
-i	Ignores the /etc/exports file and exports a directory not listed in the file
-u	Removes (unexport) currently listed exported directories
-au	Removes all currently exported directories

The *showmount* command can be used to determine information about the shared directories (exports) on a server and will list any exports currently shared including those listed in the /etc/exports file, if they are currently being shared on the server. The *showmount* command syntax is *showmount* [options] [server1].

The default value for server1 is the value returned by the *hostname* command. With no options, *showmount* shows the clients that have mounted directories from the host. Some of the available options are shown in Table 2.4.

To establish a connection to the share, the client can use the *mount* command. The following demonstrates the way a client can establish a NFS connection:

```
mount server1:/share /mount_point
```

This makes accessible all the files underneath the directory /share on the server1 by changing to the /mount_point directory. An automatic mounted connection to the NFS share is established at boot time by inserting a line entry to the /etc/fstab file.

Swap

Linux requires virtual memory to ensure its successful performance, which can exist as a file or as an entire partition for storage. Virtual memory is accomplished by dividing the system's physical RAM into units known as *pages* and transferring less frequently used physical units of RAM (pages) to the hard disk drive using a process known as *swapping*. When the system swaps out pages of memory to the hard disk drive, the system's RAM is freed up to perform additional functions.

Although swapping offers advantages by extending its access to more memory, pages stored and retrieved on the hard disk drive are accessed slower than pages that only resided in physical memory (RAM). For the Linux system to use swap space, a special file or swap partition must be created first. The creation of a swap file entails the creation of a special file, which must be designated to function as a swap file and is created with the *dd* command. Below is an example of the creation of empty Linux swap file.

```
dd if=/dev/zero of=/newswapfile bs=1024 count=1048576
```

Table 2.4 *Showmount* Options

***Showmount* Option**	**Purpose**
-a, --all	Uses the format *hostname:directory*, where *hostname* is the name of the client and *directory* is the mounted directory
-d, --directories	List client mounted directories
-e, --exports	Prints the servers list of exported filesystems
-h, --help	Provides help summary

The command creates a swap file named *newswapfile*. The input file */dev/zero* is a special Linux file that provides null characters. The newly created swap file is 1 GB in size.

To designate a partition or special file to be used as swap space, use the *mkswap* command. This command sets up a swap area on a partition or special file. The following commands prepare first a partition and then a special file as a swap file:

```
mkswap /dev/hdb1

mkswap /newswapfile
```

The *swapon* command is used to designate the specific devices or files on which paging and swapping is to take place. For the partition, you would need to prepare it using the *swapon* command as root:

```
swapon/dev/hdb1
```

To disable or turn off the swap files or swap partitions, the *swapoff* command can be used. The *swapoff* command disables swapping on the specified partition or file. When the *-a* flag is used, swapping is disabled on all known swap devices and files. The *swapon –s* command will display the current status.

SUMMARY OF EXAM OBJECTIVES

Linux filesystems provide the operating system with a framework for the storage, organization, modification, removal, and retrieval of digital information. Filesystems are responsible for organizing and maintaining files, folders, metadata, and residual data as containers for storing digital information. In addition, you learned that different filesystem types are available for local and network access (for example, SMB and NFS).

You learned about the mounting and unmounting of filesystems using the *mount* and *umount* commands. Automatically mounted filesystems are defined in the /etc/fstab file. After creating a partition, a filesystem must be assigned to the filesystem. The Linux *mkfs* command assigns filesystem to the partition. The Linux filesystem is a hierarchical structure used to organize directories (folders) and files and is based on the FHS. Filesystem management described the tools needed to manage local and remote filesystems. Commands to determine the amount of disk space used and remaining (for example, *du* and *df*) and the various commands for implementing and managing disk space limitations for users (for example, *edquota*, *quotacheck*, and *repquota*) were discussed. Additionally, the repairing of corrupted filesystems using *fsck* command, the mounting of unique loopback filesystems, the accessing of remote NFS filesystems, and the preparation of swap files or partitions using the Linux commands *mkswap*, *swapon*, *swapoff*, and *swapinfo* were described.

TOP FIVE TOUGHEST QUESTIONS

1. You need to use *fdisk* to establish a partition for a new SCSI disk drive you want to add for extra storage space. The original drives are all IDE drives. Which is the correct syntax?
   A. *fdisk /dev/SCSI1*
   B. *fdisk /dev/IDE*
   C. *fdisk /dev/sda*
   D. *fdisk /dev/sdb*

2. Which file, when the system initially starts up, will automatically mount filesystems?
   A. /etc/fstab
   B. /boot/fstab
   C. /dev/devices.map
   D. /etc/grub.conf

3. What is an ISO loopback device?
   A. The transformation of special file into a virtual Linux filesystem
   B. A device that returns feedback tests to the monitor
   C. The /null driver device
   D. The IP address 127.0.0.1

4. Which Linux command is used to designate a specific file or partition for swapping?
   A. */swap*
   B. *fileswap*
   C. *swapon*
   D. *GRUB*

5. What is the purpose of the Linux *exportfs* command?
   A. To function as the Linux bootloader
   B. To partition a storage device
   C. To designate a specific file or partition for swapping
   D. To activate the access of shared NFS directories

ANSWERS

1. Correct answer and explanation: **C**. Answer **C** is correct; the SCSI device notation for the first disk is /dev/sda.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answers **A** and **B** are incorrect because both do not exist as valid default Linux device names. Answer **D** is incorrect; the SCSI device notation for the second SCSI disk is /dev/sdb. The system indicated that this is the first SCSI drive added.

2. Correct answer and explanation: **A**. Answer **A** is correct; the /etc/fstab file is used to define and automatically mount filesystems.

Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect because the file does not exist. Answers **C** and **D** are files used during the loading of the Linux Kernel by GRUB.

3. Correct answer and explanation: **A**. Answer **A** is correct. The Linux operating system offers support for an additional unique type of filesystem known as the *loopback device*. This allows the mounting of an ISO image, so you can view/edit the files before burning a CD or DVD.

Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect because the Linux echo command will return feedback to the monitor. Answer **C** does not exist. Answer **D** is the network adaptor loopback address for testing network connectivity.

4. Correct answer and explanation: **C**. Answer **C** is correct; *swapon* is the Linux command used to designate a specific file or partition for swapping.

Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect; / swap is a precreated swap partition created during the initial installation. Answer **B** is incorrect because it is an invalid Linux command. Answer **D** is incorrect; GRUB is the Linux bootloader program.

5. Correct answer and explanation: **D**. Answer **D** is correct; *exportfs* is the Linux command used to activate the access of shared NFS directories on the NFS server.

Incorrect answers and explanations: **A**, **B**, and **C**. Answer **A** is incorrect; GRUB is the Linux bootloader program. Answer **B** is incorrect; *fdisk* is the Linux command used to partition a storage device. Answer **C** is incorrect; *swapon* is the Linux command used to designate a specific file or partition for swapping.

CHAPTER 3

Booting Linux

Exam objectives in this chapter

- GRUB
- Runlevels
- Troubleshooting Boot Issues

INTRODUCTION

The Linux boot process starts when the system is powered up and the hardware is checked by the system BIOS and then determines what device (for example, floppy disk drive, hard disk drive, CD/DVD drive) will be used to boot the system. This first sector of the boot device contains the Linux bootloading program (for example, LILO, GRUB, NTLOADER), although we will describe the Grand Unified Bootloader (GRUB) program, which we will use throughout this book. The system BIOS loads GRUB into memory and executes the program and system control is now transferred to the bootloader. The various processes are described in the following sections.

EXAM WARNING

For the Linux+ exam, the Linux booting process is based on the Intel x86 CPU architecture. Linux also supports the booting of other CPU hardware architectures (for example, AMD, Alpha, ARM, IA-64, m68k, MIPS, PA-RISC, PowerPC, S/390, SPARC), but the boot processes are different.

GRUB

GRUB is used for starting modern Linux operating systems and is the first program on any storage device that the computer executes. The purpose of the bootloader program is to perform a sequence of events on your computer to load the main operating system. In essence, the bootloader receives control from the system BIOS process, performs a sequence of events, and then transfers control to the operating system kernel.

Most modern bootloaders are dynamically configurable and can be executed during and after the system has been booted. The bootloader can load predefined configuration files during startup or can support boot-time changes (for example, selecting different kernels, virtual file systems) via a boot command line prompt. In addition, the bootloader application can make modifications to boot-time configuration files and test the files prior to using them after control has been transferred to the operating system kernel.

Installing GRUB and Booting Linux

Prior to the execution of GRUB, the system BIOS loads into memory the Master Boot Record (MBR) and executes it contents. The total size of the MBR is 512 bytes, which contains the bootloader program and disk partitioning information. The preinstallation form of the GRUB program is divided into two stages and the MBR loads GRUB stage 1. The stage 1 program uses the first 446 bytes with the remaining 64 bytes allocated to the partition table of the hard disk drives. The purpose of GRUB stage 1 is to find and load GRUB stage 2 (which may reside physically elsewhere on the hard disk). GRUB stage 1 must be flexible enough to access many different file system types. This flexibility is accomplished because GRUB stage 1 has loaded a large number of mass storage device drivers. Once loaded, GRUB stage 2 can perform the following three different functions:

- load a predefined Linux kernel (for example, vmlinuz-version.gz).
- allow selection of which operating system to boot on dual boot system.
- entry of different boot parameters.

Once GRUB stage 2 has loaded the Linux kernel, it must also load a virtual file system and execute the Linux kernel.

Fast Facts

Linux is started using a bootloader, often using GRUB which can be summarized as follows.

- Is loaded by the BIOS and resides in the MBR.
- Can access many different filesystems (ext3, file allocation table (FAT), VFAT, and so forth).
- Is dynamically configurable after you have installed Linux.
- Main configuration file is /etc/grub.conf.
- GRUB boot menu is /boot/grub/menu.1st.

GRUB Configuration Files and Commands

GRUB is a dynamically configurable bootloader application and allows for postinstallation changes to the system for ensuring a successful boot-up process, if

device changes or Linux kernel modifications are required. The GRUB application allows for alterations to three important configuration files.

The first file, */etc/grub.conf*, contains information about the disk partition used to find and load GRUB stage 2, as shown in Figure 3.1. This file instructs GRUB stage 1 where to look for the GRUB stage 2 image (/boot/grub/stage2) for loading.

The second file, */boot/grub/menu.lst*, is a file that functions as the GRUB Boot Menu. It contains content about the partitions and operating systems that can be booted, the loading of different kernels, the establishing of a different default kernel, and various other boot option modifications, as shown in Figure 3.2. The Troubleshooting Boot Issues section, presented later in this chapter, provides GRUB prompt procedures for entering commands that can be issued to dynamically modify the Linux kernel loading and *runlevel* processes.

The /boot/grub/menu.lst file options used for modifying and/or selecting a different kernel and various other functions are presented in Table 3.1

```
linux-01vc:/etc # cat grub.conf
setup --stage2=/boot/grub/stage2 --force-lba (hd0,1) (hd0,1)
quit
linux-01vc:/etc #
```

FIGURE 3.1
/etc/grub.conf File

```
linux-01vc:/boot/grub # cat menu.lst
# Modified by YaST2. Last modification on Mon May 11 12:50:48 UTC 2009
default 0
timeout 8
##YaST - generic_mbr
gfxmenu (hd0,1)/boot/message
##YaST - activate

###Don't change this comment - YaST2 identifier: Original name: linux###
title openSUSE 11.1 - 2.6.27.7-9
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.27.7-9-pae root=/dev/sda2 resume=/dev/sda1 splash=silent showopts vga=0x317
    initrd /boot/initrd-2.6.27.7-9-pae

###Don't change this comment - YaST2 identifier: Original name: failsafe###
title Failsafe -- openSUSE 11.1 - 2.6.27.7-9
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.27.7-9-pae root=/dev/sda2 showopts ide=nodma apm=off noresume nosmp maxcpus=
off powersaved=off nohz=off highres=off processor.max_cstate=1 x11failsafe vga=0x317
    initrd /boot/initrd-2.6.27.7-9-pae

###Don't change this comment - YaST2 identifier: Original name: floppy###
title Floppy
    rootnoverify (fd0)
    chainloader +1
linux-01vc:/boot/grub #
```

FIGURE 3.2
/boot/grub/menu.lst File.

Table 3.1 /boot/grub/menu.lst Options

/boot/grub/menu.lst Options	Purpose
default	This option is used to instruct the system to use the designated Title entry to boot by default. Examples include: ■ default 0 – for the first menu ■ default 1 – for the second menu ■ default 2 – for the third menu
timeout	This option is to instruct the system to immediately boot the default selection or wait a prescribed amount of time. Examples include: ■ timeout 5 – means wait 5 s before automatically booting the system. ■ timeout 10 – means wait 10 s before automatically booting the system. ■ timeout 0 – means boot the default selection immediately.
title	This option indicates the setting displayed by the boot-menu title. Examples include: ■ title Linux ■ title Failsafe ■ title GNOME User Interface
root	This option provides a device or partition name indicating the location of the kernel and initrd files. Examples include: ■ root (hd0,0) – this represents the first hard drive and the first partition. ■ root (hd0,1) – this represents the first hard drive and the second partition.
kernel	This option presents the location and name for the Linux kernel (for example, vmlinuz). This is the option used to select a different kernel to boot the system. It also specifies the default runlevel by placing the runlevel number at the end of the line. Examples include: ■ kernel/vmlinuz-*version* ■ kernel/boot/vmlinuz-*version*
initrd	This option provides the name and location of the virtual file system. This is the option used to select a different virtual file system. Examples include: ■ initrd/boot/initrd ■ initrd/initrd

(*Continued*)

Table 3.1 (*Continued*)

/boot/grub/menu.lst Options	Purpose
root = */disk partition*	This option instructs the system where to mount the Linux root (/) directory. This is the option to use if a different device and/or partition is used for the root (/) directory. Examples include: ■ root = /dev/sda2 ■ root = /dev/sdb1
showopts	This option is used to display parameters listed after this option on the boot screen. Examples include: ■ showopts acpi = off ide = nodma

The final configuration file, */boot/grub/device.map*, is a unique file whose purpose is to map the Linux device names to the GRUB/BIOS device naming conventions, shown below:

```
(fd0)/dev/fd0
(hd0)/dev/sda
```

GRUB is also an executable program (known as GRUB Shell) accessible from the root prompt and offers the following:

- change the disk order
- view other boot loaders
- view hard disk partition details
- modify partition settings
- boot user-defined configuration files
- password protect the system during the bootloading process

Using this command, you can install or test GRUB configuration settings before applying the modifications to the system during the next boot process. To implement changes made to the GRUB configuration device.map file, from the Linux prompt, execute the following command to reload device.map and execute commands listed in the grub.conf file.

```
grub -batch</etc/grub.conf
```

DID YOU KNOW?

GRUB is a bootloader for Linux and also an executable shell command to configure the bootloader from the root prompt. You can test new boot time configuration settings prior to applying them. This is useful if you need to modify the partition details or to boot new configuration files.

In rare occasions, you may be required to reinstall the GRUB preinstallation application to the hard disk on a running system. To accomplish this task, you can use the *grub-install* command, as shown in Figure 3.3. This command installs GRUB to either the MBR or another partition and checks for errors.

```
grub-install/dev/sda
```

RUNLEVELS

This section introduces the functionality of the init program and the seven different runlevels.

The *init* Command

The *init* command is responsible for executing runlevels and is the last step in the boot process. It will be identified by the process id of 1. It is responsible for starting system process defined in the */etc/inittab* file.

Runlevels

Runlevels are specialized scripts that define how a computer system starts or stops by executing various services or processes and the level of root administration and user access. The runlevel is changed by having a privileged user run *teleinit* or *init*, which sends appropriate signals to *init*.

Seven different runlevels exist, numbered from 0 through 6 although runlevel 4 is not often used.

Fast Facts

Knowledge of the runlevels and what each is designed to do is very important and the most important are described as follows:

- 0 terminates all programs and shuts down the system.
- 1 is for single-user mode; s or S can also be used.
- 2 is for multiuser mode without network connectivity.
- 3 is for multiuser mode with network connectivity and starts in command line mode.
- 5 starts multiuser mode with network connectivity and starts the graphical user interface (GUI).
- 6 reboots the system.

Most systems set their default runlevels to either 3 or 5.

To execute runlevels, two types of scripts are located in the /etc/init.d directory called through symbolic links. The first set of scripts, executed by the *init*

```
linux-01vc:/home/LinuxExpert1 # grub-install /dev/sda


    GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word, TAB
   lists possible command completions.  Anywhere else TAB lists the possible
   completions of a device/filename. ]
grub> setup --stage2=/boot/grub/stage2 --force-lba (hd0,1) (hd0,1)
 Checking if "/boot/grub/stage1" exists... yes
 Checking if "/boot/grub/stage2" exists... yes
 Checking if "/boot/grub/e2fs_stage1_5" exists... yes
 Running "embed /boot/grub/e2fs_stage1_5 (hd0,1)"... failed (this is not fatal)
 Running "embed /boot/grub/e2fs_stage1_5 (hd0,1)"... failed (this is not fatal)
 Running "install --force-lba --stage2=/boot/grub/stage2 /boot/grub/stage1 (hd0,1) /boot/grub/stage2 p /boot/gr
ub/menu.lst "... succeeded
Done.
grub> quit
linux-01vc:/home/LinuxExpert1 #
```

FIGURE 3.3
Grub-install /dev/sda.

command, is started during the system's boot process or whenever you initiate the shutdown process. These scripts are contained in the /etc/inittab file, as shown in Figure 3.4. The default runlevel setting is located in the file /etc/inittab. The default runlevel entry looks like the following in the file:

```
id:5:initdefault:
```

For the file presented in Figure 3.4, the runlevel is configured to startup in multiuser mode and provide network connectivity and X Windows support for Windows Managers. The file also indicates what subdirectory to search for specific services or programs to execute. For the default runlevel, /etc/init.d/rc5.d is the directory containing default services and programs scripts. Each of the runlevel folders contain scripts that are executed when a runlevel is started and stopped. For files containing scripts used for starting a runlevel, these files begin with the capital letter "S." For files containing scripts used for stopping a runlevel, the files begin with the capital letter "K." The second sets of scripts are executed whenever the runlevels are changed from one runlevel to another runlevel. The /etc/init.d/rc file is called to ensure the scripts are executed in the proper sequence.

TROUBLESHOOTING BOOT ISSUES

This section provides an overview of different options available to resolve Linux boot issues. *dmesg* is used to send kernel messages to a standard output (for example, console). The kernel can send messages to the computer monitor representing hardware devices detected and configured during startup. *dmesg* accomplishes this by being able to print or control the kernel ring buffer and can assist in troubleshooting or obtaining information about system hardware. The *dmesg* syntax is as follows:

```
dmesg [-c] [-n level] [-s bufsize]
```

```
LinuxExpert1@linux-01vc:/etc> cat inittab
#
# /etc/inittab
#
# Copyright (c) 1996-2002 SuSE Linux AG, Nuernberg, Germany.  All rights reserved.
#
# Author: Florian La Roche, 1996
# Please send feedback to http://www.suse.de/feedback
#
# This is the main configuration file of /sbin/init, which
# is executed by the kernel on startup. It describes what
# scripts are used for the different run-levels.
#
# All scripts for runlevel changes are in /etc/init.d/.
#
# This file may be modified by SuSEconfig unless CHECK_INITTAB
# in /etc/sysconfig/suseconfig is set to "no"
#

# The default runlevel is defined here
id:5:initdefault:

# First script to be executed, if not booting in emergency (-b) mode
si::bootwait:/etc/init.d/boot

# /etc/init.d/rc takes care of runlevel handling
#
# runlevel 0  is  System halt   (Do not use this for initdefault!)
# runlevel 1  is  Single user mode
# runlevel 2  is  Local multiuser without remote network (e.g. NFS)
# runlevel 3  is  Full multiuser with network
# runlevel 4  is  Not used
# runlevel 5  is  Full multiuser with network and xdm
# runlevel 6  is  System reboot (Do not use this for initdefault!)
#
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
#l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

FIGURE 3.4
Linux Inittab Scripts File.

- "-c" clears the kernel ring buffer.
- "-s bufsize" determines the buffer size to query from the kernel ring buffer. The default size is 16392.
- "-n level" determines the type of messages sent to the computer monitor. When "-n 1" is used, only panic messages appear on the computer monitor. All other messages are prevented. The *dmesg* command will not print or clear the kernel ring buffer when the "-n" option is used.

Entering *dmesg* (or *dmesg | more*) without any parameters will display the entire list of kernel messages sent to the standard output.

Crunch Time

The use of *dmesg* when you are trying to resolve boot time issues is very useful and important to remember for the exam. The command will be able to show you the hardware devices the kernel detects and if it can configure them.

The next option, kernel options, allows you to enter information to be executed by the Linux kernel during the booting of the system. The kernel parameters can be added by editing /boot/grub/menu.lst or by entering information at the boot prompt. The /boot/grub/menu.lst provides the default kernel parameters, as shown earlier in Figure 3.2. In addition, failsafe kernel parameters can be predefined that will enable Linux to boot even under problematic circumstances. To dynamically modify the kernel during the boot process with GRUB, you can enter various kernel parameters such as the modification of the runlevel parameter.

Fast Facts

The use of a Rescue System is very useful as it loads a specialized kernel without a GUI from a variety of sources.

- The original distribuition disk can be used, so you can then modify configuration files, check the file system for defects, verify and/or modify the bootloader configuration, resize the partition, and make a few other critical system modifications that may be necessary.
- You can use a Live CD that allows you to perform a system rescue by booting the system and then mounting the disk partitions to fix or repair configuration or boot issues. These can be on CD, DVD, or even a USB memory stick.

Single-user mode is used to bypass the requirement to enter a root password (many Linux distributions now force the entering of a root password). Typically, this option is used to gain access to the root prompt to change a lost or forgotten password. When the Linux system is booted under runlevel 1 (single-user mode), you will directly get a root prompt and you can execute the *passwd* command to modify the root password. As you are logged in as root, you will not be prompted to enter the old password.

SUMMARY OF EXAM OBJECTIVES

In this chapter, we discussed the Linux boot process and the information you will be required to know during the booting of a Linux system. GRUB is one of the default bootloading applications and loads the Linux kernel from the specified boot device. GRUB is used during the initial booting of the system and afterwards as an application to install or test GRUB configuration settings prior to applying the modifications to the system during the boot process. Three critical GRUB configuration files were presented: */etc/grub.conf*, */boot/grub/menu.lst*, and */boot/grub/device.map*. The *grub-install* command is used to reinstall the GRUB preinstallation application to the hard disk on a running system. This command installs GRUB to either the MBR or a partition and checks for errors.

Runlevels use a collection of scripts that define how a computer system starts or stops by executing various services or processes and the level of root administration and user access. The runlevels are numbered from 0 through 6 with the default runlevel settings for the systems located in the /etc/inittab file.

The final section, Troubleshooting Boot Issues, introduced approaches to resolve boot problems. The *dmesg* command can assist in troubleshooting or obtaining information about system hardware and is used to send Linux kernel messages to a standard output by being able to print or control the kernel ring buffer.

The dynamic modification of the kernel option allows you to enter information to be executed by the Linux kernel during the booting of your system. The kernel parameters can be added by editing /boot/grub/menu.lst, entering information at the boot prompt or by dynamically modifying the kernel during the boot process with GRUB.

The System Rescue approach starts a specialized Linux kernel without a GUI. The Linux kernel loaded can be obtained from a bootable device (for example, LiveCD). Once the Linux kernel is loaded into RAM, you can modify configuration files, check the file system for defects, verify and/or modify the boot loader configuration, resize the partition, and a few other critical system modifications that may be necessary.

TOP FIVE TOUGHEST QUESTIONS

1. You are the IT system administrator for the Linux systems in your department. You need to make changes to the default runlevel setting. Which file contains the default runlevel setting?
   A. /etc/inittab
   B. /etc/grub.boot/inittab
   C. /boot/grub/device.map
   D. /etc/init.d

2. Your IT department has made several hardware device changes. These include changes like modifications to the hard disk drives. You need to make modifications to the GRUB bootloader. Which file should you edit to configure the GRUB stage 2 image?
   A. /etc/menu.lst
   B. /boot/grub/menu.lst
   C. /etc/grub.conf
   D. /boot/grub/grub.conf

3. You are the IT system administrator for the Linux systems in your department. You need to make changes to the GRUB device naming conventions. Which file contains the default runlevel setting?
   A. /etc/device.map
   B. /etc/grub.boot/device.map
   C. /boot/grub/device.map
   D. /etc/init.d

4. The Linux kernel is a critical component in the Linux boot process. Where does it reside on the system?
   A. The /kernel directory
   B. The /grub/boot/kernel directory
   C. The /boot directory
   D. The /boot/kernel directory

5. The Linux bootloader is a very critical component in the Linux boot process. Where does it reside on the system?
   A. It resides in the Master Boot Record
   B. It resides inside the Linux kernel
   C. The /etc directory
   D. Inside the system BIOS

ANSWERS

1. Correct answer and explanation: **A**. Answer **A** is correct because the /etc/inittab contains the default runlevel setting. The default setting for runlevel 5 looks like the following:

   ```
   id:5:initdefault
   ```

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect because the directory path /etc/grub.boot does not exist. Answer **C** is incorrect because /boot/grub/device.map is used to map Linux device names to GRUB device naming conventions. Answer **D** is incorrect because /etc/init.d is the directory containing the runlevels scripts.

2. Correct answer and explanation: **C**. Answer **C** is correct because /etc/grub.conf contains directory information about the disk partition used to find and load GRUB stage2 image.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect because the menu.lst does not reside in the /etc directory and the file is used to determine which operating system is load and booted based on menu item selected. Answer **B** is incorrect because the file is used to determine which operating system is loaded and booted based on the menu item selected. Answer **D** is incorrect because the grub.conf does not reside in the /boot/grub directory.

3. Correct answer and explanation: **C**. Answer **C** is correct because /boot/grub/device.map is used to map Linux device names to GRUB device naming conventions.

   For example:

   ```
   (hd0)→/dev/sda
   ```

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect because the directory path /etc does not contain device.map. Answer **B** is incorrect because the directory /boot/grub.boot does not exist. Answer **D** is incorrect because /etc/init.d is the directory containing the runlevels scripts.

4. Correct answer and explanation: **C**. Answer **C** is correct because the Linux kernel is located in the /boot directory.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answers **A**, **B**, and **D** are incorrect because these directories do not exist.

5. Correct answer and explanation: **A**. Answer **A** is correct because the Linux Bootloader resides in the Master Boot Record (MBR). The MBR is the bootloading sector. The system BIOS reads the first sector of the boot device. This sector is 512-bytes in size. For a hard disk drive, this special sector is known as the MBR.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect because the Linux bootloader is used to retrieve the Linux kernel. Answer **C** is incorrect because /etc is a folder mounted by the Linux kernel once it obtains control from the Linux bootloader. Answer **D** is incorrect because the system BIOS reads the first sector of the boot device. This sector contains the Linux boot loading program. The system BIOS loads the bootloader into memory and executes the program.

CHAPTER 4

Configuring the Base System

Exam objectives in this chapter

- User Profiles
- Device Management
- Networking

INTRODUCTION

This chapter explains how to configure system and user profiles as well as the common environment variables, the management of the various devices and where these as located in the disk structure, and the fundamentals of Linux networking utilizing TCP/IP and how to mange this within Linux.

User management is one of the fundamental tasks that need to be understood for day-to-day management of a Linux system. Ensuring that users have the correct rights and environment setup will assure that their experience in using the system is favorable and that the support overhead is kept to a minimum.

The networking of the computer system, whether via wired or wireless connection, is usually a given necessity in today's world. The computer also typically uses Transmission Control Protocol/Internet Protocol (TCP/IP) as the transport mechanism for the network connections and the different options available to set up this will be discussed. The basics in connecting to name servers (NSs) and Dynamic Host Configuration Protocol (DHCP) servers are also discussed, and the majority of parameters are explained.

USER PROFILES

Any installation of Linux includes the creation of a number of different user accounts: the superuser, a normal day-to-day user, and a system user. A normal user can add, delete, and modify his or her files and those that have the appropriate attribute set. These users cannot make system-wide changes nor can they manage other users on the system.

A superuser (also referred to as a system administrator) has global privileges and can create and delete users and can change the permissions of files located within the filesystems. There is a special superuser known as root, with a user ID and group ID of 0. This user has full and unrestricted rights to manipulate any file, to traverse to any directory, and to execute any program.

Fast Facts

All users in a system will have a profile which is configurable.

- Only superusers have the rights to configure other user's profiles.
- Use export when you set environment variables if they need to be elsewhere in the environment.
- Use the full path name when you need to run commands that are not located in a directory specified in your *path* variable.
- Setting a variable may require additional actions, for example, setting the *PRINTER* variable also needs printer drivers installed.

A system user is an administrative account that is used by the system itself for the running of various administrative tasks. System users differ from other users on the system in that they do not have a home directory or password; nor can they be accessed via the normal system login prompt.

System and User Profile and Environment Variables

Users on the system are able to customize their profile to suit specific needs and preferences, held in environment variables throughout the system. There are three different shells in bash (Bourne-again shell): login shell, normal shell, and interactive shell. The login shell reads .bash_profile located in the user home directory or /etc directory, and interactive shells read ~/.bashrc. The environment variables are named objects that contain information that can be used by one or more applications.

A summary of the bash (Bourne-again shell) startup files is shown below:

- /etc/profile: The system-wide startup file and will be executed when a user logs on.
- /etc/bash.bashrc: This is often linked to /etc/bashrc and is called per interactive shell startup.

Both these files will be protected, and only the superuser (root) will be able to make changes to the file. As the files may be overwritten with a system upgrade, it is not recommended that changes are made to them directly.

Normal users have two similar files that are also called:

- /home/user/.bash_profile: A personal startup file and is executed when a user logs into a system.
- /home/usr/.bashrc: The personal interactive shell startup file.

The format for creating and modifying an environment variable within bash (Bourne-again shell) is always in the format:

```
NAME=value
```

Crunch Time

The setting of a variable only makes this available in that shell. To move the variable from the shell to the environment, the *export* command has to be used:

```
export NAME=value
```

This allows programs other than the shell to access this variable (for instance, a file editor).

PS1

When a user first logs on, he or she will be greeted by a prompt. The user prompt is defined initially in the file /etc/bash.bashrc as the environment variable *PS1* and can display a vast array of data.

Typical variables that are used:

\d Date in "Wed Sep 09" format
\h First part of the host name
\u Username
\t Time in 24-h format

PS2

The *PS2* variable is very similar to the *PS1* variable except that it is displayed when the user issues an incomplete command and the system will prompt and wait for the user to complete the command and press **Enter** again. This default secondary prompt is the > sign and can be changed by altering the *PS2* variable.

PATH

The *PATH* variable is used by commands to locate a specific command or application. When you enter a command, the shell looks in each of the directories specified in the PATH command. The *PATH* variable contains the list of directories separated by a colon, such as /bin:/usr/bin:/usr/local/bin.

EDITOR

The *EDITOR* variable defines the default text editor, for example, *ed*, *vi*, *vim*, and *emacs*. To set the default editor to be *vim*, use *EDITOR=vim* or *EDITOR=/usr/bin/vim*.

It is better to use the complete path name (especially in the .bashrc script) to ensure that the variable is defined correctly.

TERM

The *TERM* variable is to set up the type of terminal in use, which can be particularly important when using screen-orientated programs such as a text editor. The variable can also be set using the *tset* command, which is often used in the login script for users to allow them to choose the type of terminal they are logging in from.

PAGER

The *PAGER* variable controls the output to the screen, such as the *man* command. This allows the display of the output in a controlled manner. The typical values for *PAGER* are *more* and *less*. Although these are similar, less has additional features (such as scroll backwards with the *b* key) and hence is the preferable value to use as a default.

HOME

The *HOME* variable is set whenever you login to the system and will be set to /home/username (where username is your login name). This should not be changed as a lot of programs use this to create or find files in your personal home directory. In addition, the shortcut ~ references the *HOME* variable.

PRINTER

The *PRINTER* variable defines the default printer and is mainly used for command-line programs to print. The setting of the variable does not preclude the need to install the printer on the system.

DEVICE MANAGEMENT

The management of devices on a Linux system is of critical importance and is often undertaken using the command line or graphical user interface (GUI) interface. The following section concentrates on command-line programs.

Fast Facts

You can use a number of Linux commands to obtain information about the hardware installed in the system. These commands can display a wealth of information in order to enable you to debug any issues.

- *lsusb* displays USB information.
- *lspci* displays information on Peripheral Component Interconnect (PCI) devices.
- *lsmod* extracts and displays information on loaded modules.
- /sys contains files related to the kernel.
- /proc facilitates communications between the kernel and the user processes.

lsusb

lsusb lists all the USB buses on a system and displays information about any devices attached to them. The *-v* option gives verbose output.

```
$ lsusb

Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub

Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub

Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub

Bus 001 Device 005: ID 0204:6025 Chipsbank Microelectronics Co.,
   Ltd CBM2080 Flash drive controller

Bus 001 Device 003: ID 046d:c517 Logitech, Inc. LX710 Cordless
   Desktop Laser

Bus 001 Device 002: ID 413c:0058 Dell Computer Corp. Port
   Replicator

Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

lspci

The *lspci* command displays all the information regarding the PCI buses in the system and all the devices that are attached to it. This command can also be used with the *-x* option to display the initial 64 bytes of PCI configuration to see what is loaded.

```
$ lspci -x

00:00.0 Host bridge: Intel Corporation 82855PM Processor to I/O
   Controller (rev 03)

00: 86 80 40 33 06 01 90 20 03 00 00 06 00 00 00 00

10: 08 00 00 e0 00 00 00 00 00 00 00 00 00 00 00 00

20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

30: 00 00 00 00 e4 00 00 00 00 00 00 00 00 00 00 00
```

```
00:01.0 PCI bridge: Intel Corporation 82855PM Processor to AGP
   Controller (rev 03)

00: 86 80 41 33 07 01 a0 00 03 00 04 06 00 20 01 00

10: 00 00 00 00 00 00 00 00 00 01 01 20 c0 c0 a0 22

20: 00 fc f0 fd 00 e8 f0 ef 00 00 00 00 00 00 00 00

30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0c 00
```

Also, the *-b* command can be used to display the interrupt request line (IRQ) addresses as seen by the individual cards.

```
$ lspci -b

00:00.0 Host bridge: Intel Corporation 82855PM Processor to I/O
   Controller (rev 03)

00:01.0 PCI bridge: Intel Corporation 82855PM Processor to AGP
   Controller (rev 03)

00:1d.0 USB Controller: Intel Corporation 82801DB/DBL/DBM (ICH4/
   ICH4-L/ICH4-M) USB UHCI Controller #1 (rev 01)
```

lsmod

The *lsmod* command extracts all the information about loaded *modules*, derived from the /proc/modules file, and displays it as shown below:

```
$ lsmod

Module                  Size   Used by

nls_iso8859_1           3768   1

nls_cp437               5432   1

vfat                    9764   1

fat                     46376  1 vfat

usb_storage             86620  1

ip6t_LOG                6212   7
```

/sys

The /sys directory contains all the files related to the kernel, firmware, and other system-related files. There are a number of directories under /sys to ensure that it is a well-organized structure. The overall structure can be seen using the *ls* command.

```
$ ls -l /sys

total 0
```

```
drwxr-xr-x   2 root root 0 2009-05-23 10:53 block
drwxr-xr-x  16 root root 0 2009-05-23 08:47 bus
drwxr-xr-x  39 root root 0 2009-05-23 08:47 class
drwxr-xr-x   4 root root 0 2009-05-23 10:56 dev
drwxr-xr-x  10 root root 0 2009-05-23 08:47 devices
drwxr-xr-x   5 root root 0 2009-05-23 08:47 firmware
drwxr-xr-x   3 root root 0 2009-05-23 08:47 fs
drwxr-xr-x   6 root root 0 2009-05-23 08:47 kernel
drwxr-xr-x 127 root root 0 2009-05-23 10:53 module
drwxr-xr-x   2 root root 0 2009-05-23 08:47 power
```

/proc

The /proc filesystem is a virtual filesystem, which facilitates communication between the Linux kernel and the user processes. The /proc filesystem contains a number of directories that can organize the data below it and virtual files. Virtual files can be read from or written to the /proc filesystem as a method of communicating with specific entities in the kernel. Virtual files can share data from the user to the kernel or vice versa. There are a number of interesting files in the /proc filesystem such as *cpuinfo*, which identifies the type and speed of the processor installed in the system, and *modules*, which identifies the currently loaded modules in the kernel. A typical listing of the directory is shown in Figure 4.1.

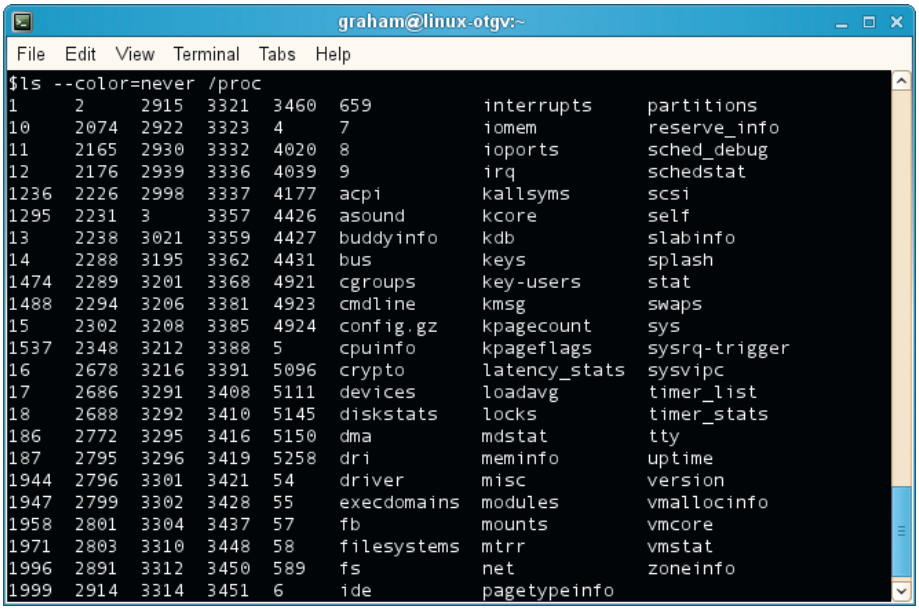

FIGURE 4.1
Example Listing of the /proc directory

The listing shows a series of numbered directories at the left-hand part of the screen for a process that is running on the computer. The directory labeled 1 is for the first process initiated: the *init* process.

DID YOU KNOW?

Although the list of compatible hardware is large and often configured correctly on installation, understanding how Linux configures devices is essential to support and maintain a system. In particular:

- Use *lsusb* and *lspci* to interrogate and list hardware devices.
- Loaded modules are listed using *lsmod*.
- *modprobe* can load and unload modules.
- /sys contains files related to the kernel and firmware.
- /proc is a virtual filesystem to communicate between the kernel and the user processes.

modprobe and modprobe.conf file

Devices can be detected during installation or when a new device is installed in a system. There are also a number of ways to initialize the hardware manually.

- *modprobe* is the high-level handler for all modules, and it can be used to unload or load a new device's kernel module.
- The /etc/modprobe.conf.local file can be edited to prompt the system to recognize and support the new hardware upon reboot.

The base system uses the /proc/modprobe.conf file to load the modules, which should not be modified. This file appends the /etc/modprobe.conf.local to itself via an include statement. Once an entry has been added to the /etc/modprobe.conf.local file and a reboot undertaken, the system performs a module dependency check.

As root, you can also manually load (and unload) a device's kernel module using *modprobe*. This command will look in the /usr/lib/[kernel version] for all the modules and files except for the optional /etc/modprobe.conf configuration file and /etc/modprobe.d directory. If the module does not exist, *modprobe* generates an error. As *modprobe* does not do anything to the module itself, all dependencies and the resolving of symbols are handled by the kernel itself. Kernel messages generated by a module failure will have to be displayed using the *dmesg* command.

Each module may need one or more additional modules loaded to enable it to function correctly, and *modprobe* will check for these dependencies in the modules.dep file, which is itself generated by the command *depmod*. The modules.dep file is located in the /lib/modules/'uname directory.

/etc/modules.conf configuration file

The behavior of *modprobe* can be altered by the optional /etc/modules.conf file. This file consists of a set of lines, which looks similar to a shell script. The file will typically only exist if you are installing kernel modules, which are not compiled directly into the kernel and not handled by *modprobe* elsewhere.

NETWORKING

It is very rare that a modern computer system is not connected to some form of network, and this requires the setup of a network card and TCP/IP on your system.

Configuring the Interface

If you are installing a network interface card (NIC) or other hardware module, ensure that it is on the supported device list for the Linux kernel you are installing. Most NICs will have one or more lights to indicate whether it is working and connected correctly. If these indicate an error, use any diagnostics tools that are available to check out the device.

When you have installed the NIC and are confident that it is working, the NIC needs to be configured. The two key commands required when configuring a NIC are *ifconfig* and *ifup*. It should be noted that superuser privileges will be required for some of the options in the *ifconfig*, *ifup*, and *ifdown* commands. The *ifconfig* command is very useful in displaying the status of a NIC, with part of the output shown below:

```
$ /sbin/ifconfig
eth0    Link encap:Ethernet HWaddr 00:0D:56:E7:9D:B1
        inet addr:192.168.1.38 Bcast:192.168.1.255
Mask:255.255.255.0
        inet6 addr: fe80::20d:56ff:fee7:9db1/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:8773 errors:5 dropped:0 overruns:0 frame:5
        TX packets:2722 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:5185021 (4.9 Mb) TX bytes:226856 (221.5 Kb)
        Interrupt:11
```

A number of actions to fault find a non-working network card are listed below. With multiple NICs, the cards are sequentially numbered eth0, eth1, and so on, and you must be sure you are manipulating the correct device.

The NIC must be turned on for the system to recognize it, and you can achieve this using *ifconfig* or *ifup* as shown below:

```
ifconfig eth1 up

ifup eth1
```

Reconfiguring the IP and subnet mask of the device may require the NIC to be inactive or to be turned on and off. This can be achieved using *ifconfig* or *ifdown* as such:

```
ifconfig eth1 down

ifdown eth1
```

EXAM WARNING

Remember that *ifconfig*, *ifup*, and *ifdown* are usually located in the /sbin directory, which is not part of the path as defined in $PATH for most users. The full directory path may therefore be required to execute the commands.

The IP address associated with the NIC can be assigned dynamically from a DHCP server, or it can have a static address assigned to it. To set a NIC to a specific IP address, the command is

```
ifconfig eth1 10.10.10.3 netmask 255.255.255.0
```

DHCP SETUP AND CONFIGURATION

For the majority of devices, the use of DHCP will be the preferred option. The host running DHCP obtains an IP address by contacting a central server, which maintains the addresses for one or more subnets. Once the client has contacted the server, it will be assigned an IP address, subnet mask, and potentially other information such as the default route and IP address of the default NS. The information that is requested is listed in the configuration file /etc/dhclient.conf.

A command-line DHCP client, /etc/dhcpcd, can also be used to configure the network interface. The Domain name server (DNS) information obtained from the server will be written to resolvconf or, if unavailable, directly to /etc/resolv.conf. In larger networks, the DHCP server may also supply configuration data for Network Information Service (NIS) and Network Time Protocol (NTP), and these will be stored in /etc/yp.conf and /etc/ntpd.conf. These services will be stopped and started by *dhcpcd* to ensure that they are notified of the change.

For support reasons, a number of other options are useful with the *dhcpcd* command:

- **release** releases the current lease and deconfigures the interface.
- **renew** attempts to renew the lease. These two commands are often done sequentially to check the connection to the DHCP server.

- **nogateway** restricts any updating of the default route.
- **nontp** stops ntp being updated.
- **nodns** stops DNS information being updated in /etc/resolvconf.

A DHCP alternative client is *dhclient*, which runs on startup and reads the /etc/dhclient.conf file. This file contains a list of all the network interfaces that are to be configured in the current system. If the interface is configured using this client, it stores the DHCP lease in /var/lib/dhcp/dhclient.leases.

CONFIGURING A WIRELESS INTERFACE

To display the status of all the wireless interfaces on a system, use *iwconfig* with no parameters or the *iwlist* command found in /usr/sbin. The wireless statistics are obtained from the /proc/net/wireless file.

Fast Facts

To set up a wireless NIC, the main parameters to use are as follows:

- *ESSID* sets the network name to enable the user to roam across a number of access points.
- *Mode* needs to be set depending on the network topology. The common parameters are *Managed* for a network of many access points, *ad-hoc* for a point-to-point connection, and *Master* if the device will act as access point.
- *Key/enc* is used to set the current encryption key and is entered in hex notation in the form XXXX-XXXX-XXXX-XXXX or as an ASCII string with the s: prefix.

NETWORK CONFIGURATION FILES

There are a number of network configuration files in the systems that are modified automatically or manually. The files can have comments embedded in them using the # symbol.

Hosts File, /etc/hosts

The hosts file maps actual IP addresses to host names and contains both IPv4 and IPv6 addresses. For large networks, this is usually undertaken by using DNS.

Services, /etc/services

The services file maps port numbers to services. A port is specific to an application and is the communication endpoint used by the transport layer protocols in the IP suite. Applications use these port numbers to ensure that the sending and receiving systems understand which application the packet is destined for.

Name Switch Service, /etc/nsswitch.conf

The Name Switch Service is used to tell the system which service to use, with potentially a number of entries per service to allow for multiple "databases" and their lookup order to be specified. Typically, there will be entries for password, hosts, networks, and services among others. The typical entries are as follows:

- nis: Use NIS or YP (formerly called Sun Yellow Pages)
- dns: Use the Domain name server
- files: Use the local files

Resolver File, /etc/resolv.conf

The resolv.conf file will normally be constructed automatically and will never need to be changed manually. It contains the list of one or more NSs, typically obtained from the DHCP service, with the format *nameserver IP_ADDRESS*.

TCP/IP Ports

There are a number of common networking ports that are used frequently. Ports 0 through 1023 are defined as well-known ports. Registered ports are from 1024 to 49151. The remainder of the ports from 49152 to 65535 can be used dynamically by applications. A brief description of these are as follows:

- Port 20 and 21: FTP data and FTP control, respectively
- Port 22: Remote login protocol secure shell (SSH)
- Port 23: Telnet, used for accessing system remotely but is not very secure
- Port 25: Simple Mail Transfer Protocol (SMTP) used by e-mail servers
- Port 53: DNS protocol
- Port 80: Used for accessing Web servers
- Port 110: The POP service or Post Office Protocol used by local e-mail clients to retrieve mail from servers
- Port 123: NTP to synchronize time with remote time servers
- Port 143: E-mail clients can use the Internet Message Access Protocol (IMAP) to retrieve mail from servers
- Port 443: This is the Hypertext Transfer Protocol (HTTP) Secure that combines the HTTP with a cryptographic protocol, which can be used for payment transactions and other secure transmission of data from Web pages.
- Port 631: The Internet Printing Protocol (IPP) used to print to printers located remotely on the network
- Port 3306: The standard port for MySQL

These ports are defined in the /etc/services file on Linux systems.

Managing Connectivity

The descriptions above have outlined the basics of network connectivity and how to install a network card into a system. The following section builds on this knowledge and defines how to manage the connectivity between systems in a network.

ROUTING

The setup of the IP address and associated data on each of the NICs within a system is only part of the required network configuration. The system needs to know where to route packets, and this is achieved using the *route* command, located in /sbin and used after the interfaces have been set up.

Any network interface will have an IP address and a subnet mask. The IP address will be in the form 192.168.1.1 (for IPv4) or 2001:db9:0:1234:0:567:1:1 (IPv6). The subnet mask identifies how many nodes are in that network, for instance, a class C network will have 254 nodes. When a machine has to communicate with another machine, it will decide how to route these packets to it. If it is on the local network, it can do so directly. Otherwise, it has to use an intermediate router to send the packets to. The *route* command used with no parameters displays the current routing table as shown below:

```
syngress> /sbin/route
Kernel IP routing table
Destination Gateway     Genmask       Flags Metric Ref  Use Iface
192.168.1.0 *           255.255.255.0 U     1      0    0   eth0
loopback    *           255.0.0.0     U     0      0    0   lo
default     192.168.1.1 0.0.0.0       UG    0      0    0   eth0
```

The first column shows the destination IP or the host name. The default gateway for this machine is the default entry and will be where packets are sent if no specific route exists for a destination. The Genmask column defines the Netmask for that particular network. The Flags column can have a number of options, with U being the route is enabled and G specifying that the destination requires a gateway. The other notable column is Iface column. This column specifies which interface is used for that route.

The *route* command can add to the routing tables and can specify a host or a network as a destination, with the default being a host. The most common route to add is that of the default gateway such as

```
/sbin/route add default gw 192.168.1.1
```

If the interface has just been configured using the *ifconfig* command, the network may have to be added by hand

```
/sbin/route add -net 192.168.1.0 netmask 255.255.25.50 dev eth1
```

IPTABLES

iptables is a user space program primarily for system administrators and is usually installed as /usr/sbin/iptables. It is used to configure the tables, rules, and filters to control the treatment of network packets into and out of the system. It uses the *Xtables* framework, which itself is used by Netfilter. The Xtables is the kernel-level component and provides an application program interface (API) for

kernel-level extensions. The tables are associated with a number of specific kinds of packet processing. Packets are processed by the system by rules in a chain, with each rule able to send the packet to another rule if necessary. All network packets into and out of the system must traverse at least one chain.

There are three predefined chains for input, output, and forward in the table. A packet traverses the chain until a rule matches the packet and decides what to do with it (such as accept or drop a packet) or returns the rule processing to the calling chain or traverses until the end of the chain is reached. The current rules can be displayed using the *iptables –L* command and must be run as root as it requires elevated privileges. As Network Address Translation (NAT) is configured from the packet filter ruleset, this is included with iptables.

DNS RECORD TYPE AND DNS RESOLUTION

The Domain name server is used to convert names to their actual IP address using the resolver process. These servers are defined in the /etc/resolv.conf file defined above. Each NS is part of a tree structure and will have an authoritative NS for its domain, such as foo.com. Each NS may delegate parts of its zone to other NSs for convenience and speed. Starting from the root domain, the server that is the authoritative NS for a domain can be found by following the chain of delegations.

Fast Facts

Each DNS will hold the data for the domain in resource records. These records hold a single fact about that domain, with the common records defined below:

- A (address) records define the actual IP address associated with a name.
- NS (name server) records define the authoritative NS for the domain.
- MX (Mail Exchanger) records define the main server for the zone.
- PTR (Pointer) records define the real name of the host for a particular IP.
- CNAME (Canonical Name) is the alias of one name to another.
- TXT (Text) Primarily for human-readable text but can also contain machine-readable data.

More information on the DNS is covered in Chapter 8, "Installing, Configuring as a Server."

NETWORK CONNECTIVITY TROUBLESHOOTING

The following will guide the user through basic network connectivity troubleshooting. When connectivity issues arise, a systematic approach is needed to ensure a quick resolution. If the machine is newly built, it is advisable to use a network connection that is known to be fully working to ensure that the physical

connections, cable, and upstream devices such as routers, switches, and DHCP servers are fully operational.

Initially, ensure that the NIC configuration is correct and then connect the network cable. The machine should now be initialized with an IP address and relevant NS information from a DHCP server or using a static information. The *ifconfig* command, with no parameters, should be executed to display the status of the NICs. For a more comprehensive output than *ifconfig*, use *netstat*. The output is listed by *sockets* (application to application connections between two computers). The common options for the *netstat* command are shown in Table 4.1.

It is often useful to have the *netstat* command running in a separate terminal window with the *–c* command while testing is being undertaken. Additionally, there will be an entry in the Address Resolution Protocol (ARP) table, located in /proc/net/arp which primary translates IP addresses to Media Access Control (MAC) addresses or the actual hardware address embedded in every NIC.

With the machine on the network, the connections to various systems and networks can be tested. Initially, use the PING command to test the hosts loopback address as shown below, using the *–c* option to limit the number of pings to 3.

```
$ ping -c 3 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.065 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.064 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000 ms
rtt min/avg/max/mdev = 0.064/0.065/0.066/0.000 ms
```

Table 4.1 Common *netstat* Options

Option	Output
-a	Show the state of a; sockets and routing table entries
-g	Displays the multicast groups configured
-i	Shows all the interfaces configured *ifconfig*
-v	Verbose output
-s	Summary of activity for each protocol
-c	Output displayed every second; this is very useful in testing
-e	Verbose output for active connections only
-C	Displays information from the route cache

In addition, the command *hostname* can be used to display the local host name. This command can be used to display the name and IP address(es) of the host. This will further clarify whether the local IP addressing is set up correctly.

When the local machine is known to be working correctly, the command can be used to test other machines. The PING command will echo back the name and, if resolved, the IP address; otherwise an error message of "ping: unknown host" is displayed. If you use a name of a well-known server, for example, *ping linux .com* and the name is resolved, then basic NS resolution is working.

To find out more about the actual route packets take from your machine to the target, *traceroute* can be used which is located in /usr/sbin. The first part of the output to the Syngress Web server is shown below:

```
$ /usr/sbin/traceroute www.syngress.com

traceroute to www.syngress.com (145.36.40.200), 30 hops max, 40 byte
   packets using UDP

1 192.168.1.1 (192.168.1.1) 1.117 ms 0.595 ms 0.621 ms

2 * * *

3 ge-3-27-ur02.grant.tx.houston.comcast.net (68.85.250.25)
   7.015 ms 7.898 ms 7.332 ms

4 te-8-1-ar01.royalton.tx.houston.comcast.net (68.85.244.101)
   10.504 ms 10.304 ms 9.740 ms

5 po-11-ar02.royalton.tx.houston.comcast.net (68.85.244.98)
   11.640 ms 11.836 ms 11.808 ms

6 po-17-ar02.greenspoint.tx.houston.comcast.net (68.85.244.130)
   13.299 ms 13.271 ms 13.276 ms

7 te-0-1-0-4-cr01.dallas.tx.ibone.comcast.net (68.86.91.57)
   17.153 ms 17.074 ms 16.860 ms

8 64.132.69.249 (64.132.69.249) 16.837 ms 16.650 ms 16.243 ms
```

Along a network there will be a number of routers which interconnect different networks together. The routers along the network decide where to send the packet, that is, to forward it to one of its interfaces. If the router does not find a matching route for the packet, it will be sent to its default route and so on until the packet reaches its destination.

Earlier in this section, you learned that when a system does not seem to recognize a name but works perfectly with IP addresses, then there is an issue with the name resolution. First, if possible, discover if it is a global issue with all your machines on the network by utilizing the *ping <server name>* command on another machine. If this works, then the problem is with your machine setup. Check that there is a valid NS defined in the /etc/resolv.conf file and that you can

traceroute to that server. If you cannot perform basic routing to these servers, then name resolution will not occur.

The Domain Information Groper or *dig* command can query the NSs listed in the /etc/resolv.conf file and then undertakes an NS query. An example of the *dig* command and output is shown below:

```
$ dig syngress.com

; <<>> DiG 9.5.0-P2 <<>> syngress.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54845
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;syngress.com.                  IN      A

;; ANSWER SECTION:
syngress.com.           300     IN      A       145.36.40.200

;; Query time: 207 msec
;; SERVER: 68.87.85.98#53(68.87.85.98)
;; WHEN: Thu May 14 14:58:26 2009
;; MSG SIZE rcvd: 46
```

It can also perform a reverse lookup, where the IP address is used instead of the name. This produces slightly different results.

```
$ dig 145.36.40.200

; <<>> DiG 9.5.0-P2 <<>> 145.36.40.200
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 1949
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;145.36.40.200.                 IN      A
```

```
;; AUTHORITY SECTION:
.           900         IN        SOA      A.ROOT-SERVERS.NET.
NSTLD.VERISIGN-GRS.COM. 2009051401 1800 900 604800 86400

;; Query time: 144 msec
;; SERVER: 68.87.85.98#53(68.87.85.98)
;; WHEN: Thu May 14 14:59:37 2009
;; MSG SIZE rcvd: 106
```

The slightly outdated command *nslookup* is still useful, in both interactive and noninteractive modes.

```
$ nslookup
> www.syngress.com
Server: 68.87.85.98
Address: 68.87.85.98#53

Non-authoritative answer:
www.syngress.com    canonical name = syngress.com.
Name: syngress.com
Address: 145.36.40.200
```

SUMMARY OF EXAM OBJECTIVES

In this chapter, you learned about how to configure the base Linux system. The basics of user and system environment variables were explained and how to configure these globally and for individual users. The main environment variables that are commonly defined were explained.

The management of devices is very important and is often complex to many individuals initially. How to add and delete modules to the kernel and where the configuration files are located is essential knowledge for any system administrators. These commands are worth experimenting with and learning more about them to ensure you can master them.

The majority of systems are networked together, and this chapter explored the basics of networking. The routing tables were explained and how to manipulate these. In modern system, it is often problems with incorrect routing that causes problems to occur. In addition, how to set up NICs with static and DHCP IP addresses was discussed, along with an introduction to DNS.

TOP FIVE TOUGHEST QUESTIONS

1. A user wants to ensure that his wireless card installed correctly in his system only connects to his company's network. This network has an SSID of mycorp and uses WPA2 for added security. How would you ensure this occurs?
   - **A.** iwconfig essid mycorp
   - **B.** iwconfig default mycorp
   - **C.** iwconfig default_ssid mycorp
   - **D.** iwconfig noroam essid mycorp

2. You wish to set up the default editor in your environment to be the *vim* editor instead of the current setting of *vi*. Which would be the best solution to achieve this?
   - **A.** Modify the */etc/env.conf* file to set the default editor environment variable *edit* to be *vim*
   - **B.** Modify the */etc/env.conf* file to set the default editor environment variable *editor* to be *vim*
   - **C.** Change the ~/*.bashrc* file to set the default editor environment variable *editor* to be *vim*
   - **D.** Change the */etc/.bashrc* file to set the default editor environment variable *edit* to be *vim*

3. A user is experiencing connectivity issues with a network port that has been working successfully for a number of months. You have tested the network by connecting another laptop to the same port, which worked. Looking at the hardware, you can see that they have an old NIC and you wish to replace it with a new one. The kernel did not recognize the NIC upon reboot. How would you add the card manually?
   - **A.** *modprobe 8139too*
   - **B.** *modprobe enable 8139too*
   - **C.** *modprobe up 8139too*
   - **D.** *add_dep module 8139too*

4. Which of the following directories is the primary location for the current hardware information of your computer?
   - **A.** /sbin
   - **B.** /proc
   - **C.** /lib/modules
   - **D.** /etc

5. Which are the following configuration files typically associated to individual user' logins with the Bourne-again shell?
   - **A.** ~/.bashrc, /etc/profile
   - **B.** ~/.bash_profile, /etc/profile
   - **C.** ~/.bashrc, ~/profile
   - **D.** /etc/.bashrc, /etc/profile

ANSWERS

1. Correct answer and explanation: **A**. Answer **A** is correct, as the *essid* option specifies the correct SSID to use.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect, as there is no default parameter. Answer **C** is incorrect, as there is no default or *ssid* option. Answer **D** is nearly correct, but there is no noroam parameter to this command.

2. Correct answer and explanation: **C**. Answer **C** is correct, as this will change the default editor to be *vim* for yourself.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answers **A** and **B** are incorrect, as env.conf is not a valid configuration file. Answer **D** is incorrect, as this is the wrong location of the .bashrc file.

3. Correct answer and explanation: **A**. Answer **A** is correct, as this is the correct notation for the *modprobe* command.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answers **B** and **C** are incorrect, as the optional parameters are incorrect. Answer **D** is incorrect, as there is no *add_dep* command.

4. Correct answer and explanation: **B**. Answer **B** is correct because the /proc directory contains the information about the various aspects of a Linux system, including hardware information such as IRQ ports and I/O address for each detected and installed device in the system.

   Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect, as the /sbin directory contains the administrative binary files, and commands such as *ifconfig* reside here. This directory does not include information about the system-only commands to manipulate the hardware. Answer **C** is incorrect in that the /lib/modules directory contains most of the hardware driver module but does not include hardware information such as the IRQ addresses. Answer **D** is also incorrect, as the /etc directory contains a number of system configuration files and installation scripts but does not contain all the hardware information.

5. Correct answer and explanation: **A**. Answer **A** is correct, as the /etc/profile configuration file contains the defaults for all users on a global basis. Each user has a .bashrc configuration in his or her home directory.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect, as not all Linux distributions have the .bash_profile configuration file. Answer **C** is incorrect, as the profile configuration file is located in /etc and not the users home directory. Answer **D** is incorrect as .bashrc is associated with global configuration files and is therefore not in the /etc filesystem.

CHAPTER 5

Using BASH

Exam objectives in this chapter

- BASH Commands
- Scheduling Tasks
- Managing Services

INTRODUCTION

The power and versatility of the command line interface (CLI) is a key strength of Linux. It is easily scripted for repetitive tasks, which can then be scheduled and implemented across multiple machines and is straightforward to manage via remote access. BASH is a command interpreter, a way for users to submit instructions to the computer.

BASH COMMANDS

If you have not installed a graphical user interface (GUI), the system will boot you directly into a command shell. Otherwise, you will have to find your systems terminal program, such as *Terminal*.

EXAM WARNING

Note that if you do not use a "monospaced" font (where every character is the same width on the screen), columns would not align correctly, and some commands use a variety of colors, making the output invisible if your background matches one of them. A pretty blue background, for example, will render all directories invisible in the default output of the *ls*' command.

Fast Facts

There are a number of basic commands that you should know:

- *pwd* (print working directory) displays the full path of your current directory.
- *ls* lists the contents of the current directory or with a path to see the contents of other directories.

- *mkdir* creates a directory.
- *rmdir* deletes a directory.
- *cd* (change directory) changes the current working directory.
- *pushd* puts the current directory on a stack.
- *popd* retrieves the topmost directory from the stack.
- *touch* will create an empty file and can also modify the access and modification times of existing files.
- *cp* will copy a file.
- *mv* will move (or rename) a file.
- *rm* will delete a file.
- A command prompt often ends with a "$" for a normal user or a "#" if you are running as "root."

Using File Commands

The commands you type into the command line are programs, and what they do can be controlled based on options you give. The general format of a BASH command is *command -option(s) parameter(s)*.

Crunch Time

Pay careful attention to capitalization as Linux distinguishes between upper and lower case, in commands, options, and file names. For example: *ls* -s shows file sizes, where *ls* -S sorts by size. Likewise, FILE1, File1, and file1 are three different files, and *ls* is a valid command, but LS is not.

The list of options that can be used with *ls* is large, with the most common ones shown in Table 5.1.

An example line from the output of *ls -l* is shown below and interpreted in Table 5.2.

```
drwxr-xr-x 7 chappel users 4096 2009-06-23 21:36 Documents
```

Table 5.1 Common Options with the *ls* Command

Option	Description
-R	List subdirectories recursively
-l	Display detail listing of a directory
-a	Do not ignore hidden files beginning with.

Table 5.2 Explanation of the *ls –l* Output

Field	Description
drwxr-xr-x	Type and permission. The first character in the type field is the file type and the next nine characters show the access rights or **mode** of the file or directory. These are described later.
7	Number of links
chappel	File owner
users	Group the file belongs to
4096	Size in bytes
2009-06-23 21:36	Last modification date
Documents	Name of the file

Table 5.3 Explanation of the Type and Permission Fields

1	2	3	4	5	6	7	8	9	10
File type	User Permissions			Group Permissions			Other Permissions		
d	r	w	x	r	w	x	r	w	x

The type and permissions field is 10 characters long; the left most character is the type of the file, and the next nine bits read, write, or execute permissions for the user, group, and others, respectively, as shown in Table 5.3.

The first character in the type and permissions field is the file type and can be

- d (directory)
- - (regular file)
- s (socket)
- p (named pipe)
- l (symbolic link)
- c (character device special file)
- b (block device special file)

The user, group, and other permissions will have one of the following five characters in it:

- r (read)
- w (write)
- x (execute)
- s (setuid; only found in the execute field)
- - (in a field means that permission is not set)

The parameters of a command can also alter the results it will give. The most common parameter variations are the wildcards "*" and "+." These are symbols that can represent multiple patterns; "*" for any number of any character and "?" for just one of any character.

There are a number of commands that can help you to look for files. *find* is very flexible, requiring a place to start, and what to look for, for example *find /home -name file 1* looks for a file named "file1" in the /home directory, then continuing to look in each subdirectory.

locate creates a database of file names that it searches through making it faster. It can only find information in its database, so changes since the last automatically scheduled database update would not be found, and it can only search on file names. The *slocate* flavor of *locate* works the same way, but it adds additional security to prevent users from searching through files they do not have access to.

whereis is intended for searching for commands and files related to them – source, binary, configurations, and help files. It does not look in user directories at all. *which* looks through your path and tells you what program you will run if you type a given command.

FILE TYPES

A proper understanding of linked files requires a little technical background information. Directories contain the names of the files that reside in them, and pointers to an *inode* for each file. The inode stores the metadata for each file – the owner, size, access rights, time of last modification, and where the data contained in each file is physically located. The extra layer of abstraction provided by the inode means that it is possible to have two different names that point to exactly the same file information. This is called a "hardlink" and is created as follows: *ln file_name hard_link_name*

The hardlink name is just as valid of a name as the original file name, and all hard links must be deleted before the space occupied by the file is freed. The nature of hard links does not permit hard links to directories or to files residing on separate drive volumes.

The soft link is an actual file, with its own inode number and contains the path to the object the file links to. Soft links are allowed to point to directories and to files on other file systems, but because they are independent files it is possible to delete the target of the soft link and leave the link file "broken," pointing to a nonexistent file. The *ls -l* command shows symbolic links with an "l" at the very first position in the row and with an arrow (→) pointing to the target file. Symbolic links are created as follows: *ln -s file_name soft_link_name*

DEVICE FILES

Anything capable of moving information in or out of your Linux system has a device file in the /dev subdirectory. This way of thinking about devices makes it easy for user programs to send or receive information; they only need to write (or read) from the appropriate file.

Block special device files move data in and out of hardware connected to the system in large chunks, and use buffers to improve efficiency, such as hard drives, USB, and tape drives. Character device files move data a single character at a time and are unbuffered. Note that not every file in the /dev directory corresponds

to a currently connected device; most are simply placeholders waiting for a new device to point to.

Named pipes FIFO (first-in first-out) are buffers that are used for communications between programs running within the computer. One program opens the pipe to write, and the other to read, and data gets transferred between them.

Special files are created using the *mknod* command, shown for character, block, and pipe files, respectively:

```
mknod new_file_name c [major_device_number] [minor_device_number]

mknod new_file_name b [major_device_number] [minor_device_number]

mknod new_pipe_name p
```

Normally, special files are created manually only when device drivers are installed manually, which should include detailed instructions with additional information.

TESTING FILES

Files frequently have an extension to help indicate what they contain, which may not be so obvious, and *file* can sort through things with a syntax *file [filename]*. An example output is shown in Figure 5.1.

test evaluates expressions, as in *test 1 -gt 2* and the answer is returned as an **exit value** that can be checked to make decisions in a script. The power of the *test* command really shows some more interesting options, such as *test -e [filename]* to check if a file exists to be sure not to overwrite it, and *test -d [directory_name]* to see if a directory exists.

The *ls -F* command lists files with various file types tagged for easier recognition. While the *file* and *test* commands are more useful within scripts, *ls -F* is more helpful for humans and classifies the file with a special character denoting what it is, such as / for directory, @ for a link file, and * for an executable file.

Editing Files Using *vi*

A good text editor is called *vi* and the enhanced version is called *vim*. The key to dealing with *vi* is understanding that it has three modes: *Command*, *Ex*, and *Edit*. When first starting *vi*, you will be in *Ex* mode, which allows you to move around in the file you are editing and to perform copy and paste commands and other advanced features like recording and replaying macros. To actually enter text, you will need to shift into *Edit* mode by typing one of the edit keys, and then go back to *Ex* mode by pressing the **Esc** key. *Command* mode is entered from *Ex* mode by typing colon (:) and is used to enter save files, enter a filename or text to be searched for. You can go back to *Ex* mode by pressing **Esc** again.

To transition to *Edit* mode, use the *i* key to insert text in front of the current cursor position, or *o* to insert text onto a new line below the line the cursor is on. Once in *Edit* mode you can type text as you normally would. Press the **Esc** key to get back to *Ex* mode.

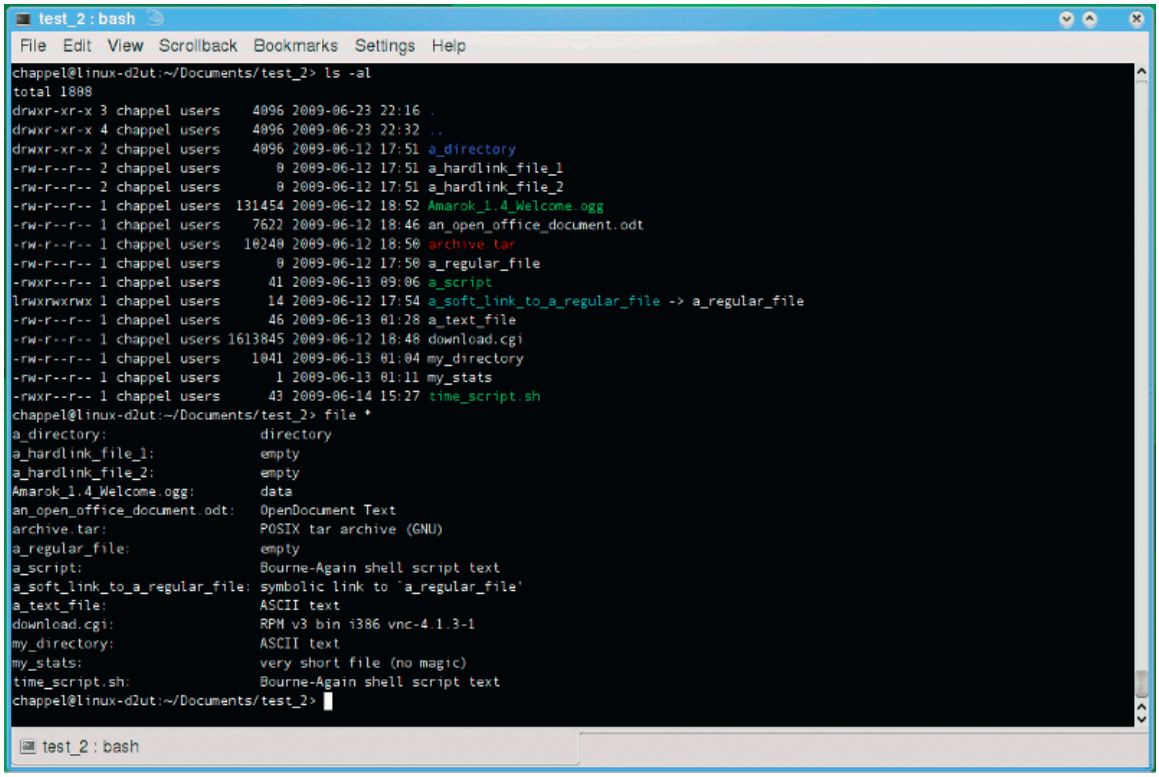

FIGURE 5.1
Example of the *file* Command

When you are finished moving and editing you can press the colon key (:) to go to *command* mode. From *command* mode you can search and replace text, save the file, and exit from *vi*.

Managing Processes

Every program, utility, or bit of code waiting in the background on your Linux machine, is called a *process*, and it is automatically handled by the kernel. Users can monitor and manage their own processes; system administrators have access to nearly all processes.

PS

ps will show you what programs are being run by your user in the terminal window it is executed in, which is usually just bash and the *ps* command itself. *ps -A* will show all the processes currently recognized by the machine. The output from the *ps* command is formatted into four columns. The first column is the process identification number or PID, which the system uses to uniquely identify each process. The next column is where the process is running – it's "control terminal," which determines where the program's input and output should go.

Next is the amount of time the processor has spent on the process. The last is the "human readable" name of the process.

Some other useful options include the following:

- *-u* or *--user* (username or user ID) will show processes associated with the given user.
- *-C* (command_name) will show processes that have the given command name.
- *p*, *-p*, or *-pid* (process_id) will show processes with the listed process ID. If *ps* is only given a number as an argument, it assumes the number is a process ID.
- *-t*, *-tty* (ttylist) will show processes on a given tty interface port. A plain "-" will show processes not associated with a port, and a *-t* without a tty number will assume you want the processes associated with the current port you are using.

KILL

kill will forcefully end a process using the process id, and *killall* will terminate a process by name instead of PID.

The *kill* command works by sending a *signal* to the process, by default the TERM signal. The syntax for signals is *kill -s [signal] PID* and common signals are shown in Table 5.4.

Signals can also be used to communicate between processes, or by the kernel to report an event or problem back to a process.

TOP

A more versatile utility is *top*, which dynamically shows running processes in real time, sorted by the amount of CPU time they are using shown in Figure 5.2.

The upper section of the *top* display shows a summary of the system, including uptime, current users (each bash session counts as a user), and various load and memory usage statistics. The lower part of the display shows the process

Table 5.4 Common Signals

Signal Name	Number	Description
Kill	9	Forcibly kill the process
INT	2	Interrupt, like typing *control-c*
TERM	15	Gracefully terminate a program
HUP	1	Often used to make the program reread its configuration
QUIT	3	Like TERM but can cause a program to copy the memory it was using to a diagnostic file

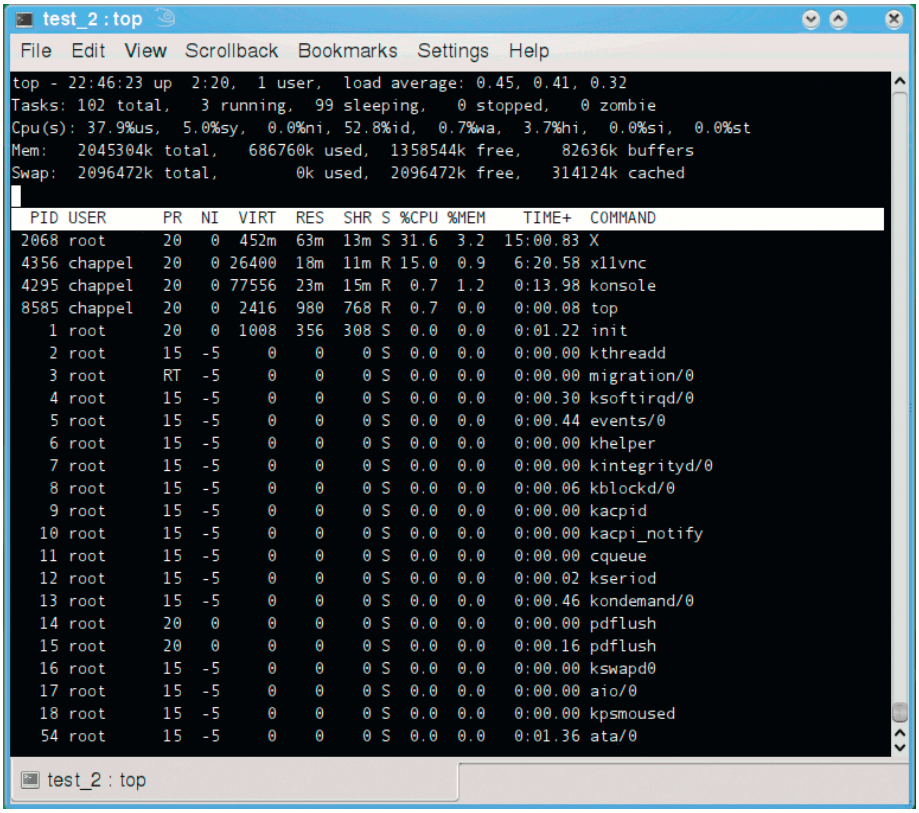

FIGURE 5.2
The *top* Command

ID, owner, amounts of various memory, and CPU resources being consumed by each process. One handy feature is a built-in *kill* option; to terminate a process, just press *k* and type in the process ID.

PSTREE

pstree shows the relationship between *parent* and *child* processes in a hierarchical tree view and *pstree -p* will also show process IDs. The init process is the parent of all the other processes and always has a PID of 1. Each child process knows its parent process ID (PPID), which is what pstree uses to trace the relationship between processes. PPID numbers can be viewed with the *ps -Al* command.

To look at not just CPU but also physical and network drive usage you can use the *iostat* command as shown in Figure 5.3.

The various CPU-related statistics shown at the top of the report are as follows:

- **%user** Percentage of processor time spent on user applications
- **%nice** Percentage of processor time spent on user applications running at a lowered priority

- **%system** Portion of processor time used by the system
- **%iowait** Percentage of time the processor has been idle when the IO system has been busy
- **%steal** Related to virtualized systems
- **%idle** Idle time

The next section of the report shows statistics for drives attached to the system, and include the following:

- **tps (transfers per second)** A transfer is considered a request for data
- **Blk_read/s, Blk_wrtn/s** Blocks read and written per second
- **Blk_read, Blk_wrtn** Total blocks read and written

NICE

A process has a priority value between –20 and +19 called a niceness value. The lower the value, the faster it runs, and niceness is inherited from the parent process. The *ps -Al* will show niceness in the NI column. The owner of a process can make it run slower, freeing up resources, and only a system administrator can make it run faster. There are two ways to adjust the niceness of a file: *nice* and *renice*. To start a program with an adjusted nice value, type the following:

```
nice -n 10 find/-name my_file.doc
```

Once the process is running it can be reset to a specific value with root access and *renice* like this:

```
sudo renice 15 [PID]
```

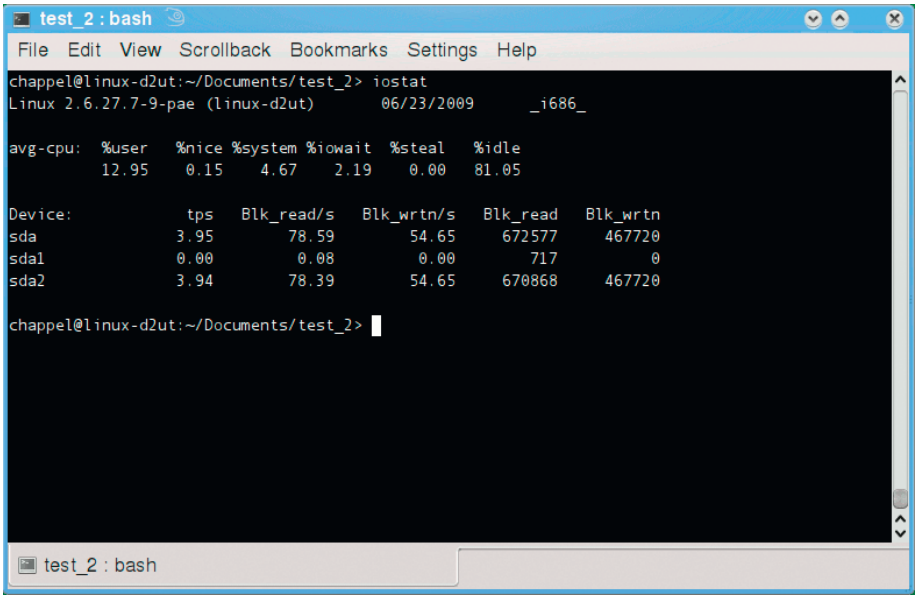

FIGURE 5.3
Example of *iostat*

Leveraging I/O Redirection

There are a number of small useful tools that are meant to work together and I/O redirection is the glue that sticks these handy little programs together. These console programs make use of three *streams* for communicating, as shown in Table 5.5.

You can use file *redirection* that allows you to use the output from one program as the input for another, or send it to a file. You can use the contents of a file as the input for a program, or use a succession of programs to create and manipulate information. Common uses of redirection are shown in Table 5.6.

EXAM WARNING

Be careful when using >, if you redirect your output stream to an existing file it will be overwritten. Use >> to append to the end of an existing file.

Table 5.5 The Three Communications Streams

Name	Number	Symbol	Normal Connection
STDIN	0	<	Keyboard
STDOUT	1	>	Console
STDERR	2	>	Console

Table 5.6 Common Uses of Redirection

Command	Result
some_program > not_the_console	Output is directed to not_the_console, which can be a file or a device
some_program < something_to_send_to_a_program	Input for some_program or file or device
some_program > data_output.txt 2> error_output.txt	Output sent to data_output.txt and errors sent to error_output.txt
some_program 2> error_output.txt	Errors sent to error_output.txt
some_program > data_and_errors.txt 2>&1 or some_program &> data_and_errors.txt	Data and error streams are combined
some_command | another_command	Sends output from one program into the input of another
ls -al | tee directory.txt	*tee* duplicates a stream and sends one copy to the display and another to a file

Some commands will only accept a certain amount of input at a time, and the solution to this is to use *xargs*, which will accept the input stream and split it into chunks to pass along a bit at a time and repeatedly running the downstream command until the input stream ends. The following example will move all of Bob's music files into his music directory:

```
find . -name "*.mp3" -u bob | xargs -i mv {} ./bobs_mp3s
```

If you put a ";" between two commands, they will run consecutively:

```
do_this_first; then_do_this; and_finally_this
```

The "==" is used to compare the equivalence of two variables, often used in scripting.

Special Devices

The /dev/null file is often called "the bit bucket," and is most often used to hide unnecessary output from a program embedded in a script. The /dev/random file is the interface to the systems random number generator, and it will produce random numbers to create cryptographic keys often using noise from various device drivers to maintain an "entropy pool" to generate the numbers. Applications that can get by with psuedorandom numbers can use /dev/urandom. The /dev/zero file works like /dev/null if you send data to it, but you can use it for an input stream to create files full of zeros.

Using System Documentation

Linux offers many ways to find additional information for all of its commands. The quickest and most basic assistance comes from using the help option of any given command – usually both *-?* and *–help* work.

EXAM WARNING

Remember that when you are taking the Linux+ exam you will have to *know* about the Linux documentation systems, but unfortunately you would not have *access* to them.

DID YOU KNOW?

There are five main ways to find information on commands:

- *man [command]* will display a brief description of the command from information in /usr/share/man or /usr/share/doc. *manpath* lists the directory where *man* looks for information.
- *apropos* searches man pages to find what you want, for example, *apropos search* shows all the entries that have *search* in their description.
- *whatis* displays one-line descriptions of a command from the man page.

- *apropos* and *whatis* use a database created by *mandb*.
- The database created by *mandb* may be handled by *makewhatis* on some distributions.
- *info* is intended as an improved way to manage documentation for all the GNU utilities and provides information in threaded chunks called *nodes*.

Accessing Kernel and Architecture Information

Linux uses a virtual file system located at /proc to represent activity within the computer, and it allows communication with various kernel and driver components. If you compare *ps -A* with *ls /proc*, you will notice that every process has its own subdirectory under /proc. There are also files for system hardware: *cat /proc/cpuinfo* tells you everything you could ever want to know about your processor and /proc/version knows all about the installed version of Linux. A slightly different way to check on your installed version is *uname -a*. Note that /proc/version shows the actual distribution name, whereas *uname -a* will show if you have a 32 or 64 bit processor.

All the settings in the proc/sys subdirectory tree can be adjusted usually using *sysctl*. Changes are lost when the system reboots, and must be added them to /etc/sysctl.conf to make them permanent. Use *sysctl -p* to get the system to reread the sysctl.conf file without having to reboot.

Basic Scripting

A script is a plain text file containing a collection of command line programs and options that can be all run as a group. Each line is read one at a time and interpreted, which can make scripts slow. Scripts need to have the *execute* permission bit set unless the *sh* command is used, which makes the *sh* program executable and the script just a parameter.

A bash script file expects to have a first line of *#!/bin/bash* and this script would like to be run with the bash shell located in the /bin directory.

Using Shell Features

The bash shell provides a history feature that remembers what commands you have previously entered, stored in a hidden file in your home directory: .bash_history and the entire list are viewed by typing **history**. Typing **!!** will repeat the last command. When you type **history** each command line is preceded by a number; typing **!<command_number>** replays the command corresponding to that number. Another handy command line feature is tab completion. Bash will try and guess what you are attempting to type if you press the **Tab** key. If you already have enough characters to uniquely identify the next word, bash will just fill it in for you. If you have not typed enough of a word, a second press of the **Tab** key will display a list of possible matches for you.

SCHEDULING TASKS

In the following sections, you will learn how to put routine tasks on an automated schedule using *cron* and schedule ad hoc tasks using *atq*.

cron (cron allow, cron deny)

For the routine tasks that need to be performed on a regular basis or at predefined intervals, there is a scheduling service *cron* that lets you put these tasks on a schedule of your own design which runs in the background, checking every minute to see if there is anything scheduled to run. The /etc/cron.allow file lists the ONLY users able to edit *cron* configuration (*crontab*) files. If that file is empty the system looks for users in the /etc/cron.deny file, which lists users barred from editing cron files. The root user is not affected by either.

crontab Command Syntax

crontab files (cron table) tells cron what to run and when to run it and is stored for users in /var/spool/cron, with the crontab name matching the username. The administrators' files are kept in /etc/crontab, and there is a /etc/cron.d directory that programs can use to store their own schedule files.

Fast Facts

crontab schedule files are edited using *crontab –e* with each line having six or seven fields and specifying one command.

- The first five fields on each line tell which minute, hour, day, month, and weekday to run a command.
- Next specifies whose user rights to run the command under.
- Lastly is the command itself.
- Valid options for the schedule fields are a * (match anything), exact number, two numbers with a dash between them for a range of values, or a list of values separated by commas to match each single value.
- *crontab -l* displays the contents of your crontab file.
- *crontab -r* deletes it.

atq

If you just want to have something run a bit later, use the *at* command. It accepts commands from either a file (*-f*) or standard input, ending with a <ctrl> *d*. A single **Ctrl + d** is sufficient if there is nothing else on the line. It takes two **Ctrl + d**'s if there is more text on the line. Use *atq* to see what jobs are queued up, and their job numbers. To cancel a job, use *atrm* followed by the job number you want canceled.

MANAGING SERVICES

As a system administrator, one of your primary jobs is to manage services running on your systems.

Fast Facts

Services, or daemons, are just programs that start and run in the background and provide services for users.

- Services are started when their assigned runlevel is initiated.
- Scripts that start each service are in etc/init.d.
- The scripts can be used manually with parameters of *start*, *stop*, or *restart*.

inetd and *xinetd*

To conserve resources, some smaller and less frequently used network-based services get bundled together in a *super server* service, which waits in the background until one of the services is needed and then loads it. This method prevents the service from just sitting there taking up memory until it is actually needed. There are two super server daemons in Linux – inetd and the newer, "extended" xinetd, which adds some features and enhanced security. Changing the services managed by inetd or xinetd is done through GUI tools or manually by editing the appropriate configuration file (/etc/inetd.conf or /etc/xinetd.conf) and restarting the service.

chkconfig

chkconfig is a handy tool for managing which runlevels a service runs in some Linux distributions. It can be used to view or change the runlevels that a service will run in. Type **chkconfig** to view the status of all services at the current runlevel.

SUMMARY OF EXAM OBJECTIVES

In the BASH and command line tools section, we covered enough to get you started creating, editing, moving, and deleting files and directories, as well as using some of the tools that are available to find files within a directory structure. We also covered using *man* and *info* to find more information about command line utilities. We even learned about linking commands together to get even more functionality out of them, and touched on how scripts work.

In the Task Scheduling section, we learned how to get programs to run automatically on a regular schedule with *cron* or just at a later time with *at*. In the Managing Services section, we learned how to tell Linux which services to run

depending on the current runlevel, and how to start, restart, and stop services using the scripts in /etc/init.d.

TOP FIVE TOUGHEST QUESTIONS

1. You are documenting your system, and want a file named *user_dirs* that contains a current list of all the user home directories. What is a quick way of doing this?
   A. echo /home > user_dirs
   B. ls /home > user_dirs
   C. ls /home >> user_dirs
   D. cp /home >> user_dirs

2. You want to check that the NTP process is really running and that your computer clock is not slow. What command will confirm that ntp is running?
   A. ps -A | grep ntp
   B. ps -C ntp
   C. ps ntp
   D. /usr/init.d/ntp status

3. You have a script named some_program that needs to start in 30 min, and you decide to try the *at* command instead of setting the alarm on your watch to remind yourself. What is the correct syntax?
   A. at 30 <return> some_program <return> <ctrl>d
   B. at now + 30 minutes <return> some_program <return> <ctrl>d
   C. at 30 minutes some_program <ctrl>d
   D. at now + .5 hours <return> some_program <return> <ctrl>d

4. You are copying a bunch of files from ./temp to ./new_stuff, but accidentally type **cp temp/* new-stuff** instead of **new_stuff**. You have been reading up on the command history function and want to use that to reenter the command correctly. What do you type?
   A. ! -change new-stuff new_stuff
   B. !!s/new-stuff>new_stuff
   C. ^-^_
   D. history -r new-stuff new_stuff

5. You just made a new script *my_new_script* and you want to run it. You remember to give an explicit path to it when you execute it by typing *./my_new_script*, but you only get an error saying you do not have permission to run the file. You remember that you need to give yourself execution rights to the file. How do you do that?
   A. chmod 744 my_new_script
   B. chmod 666 my_new_script
   C. chmod u+w my_new_script
   D. chmod g+x my_new_script

ANSWERS

1. Correct answers and explanations: **B**. Answer **B** is the correct because it will list the contents of the /home directory and that listing will be placed in the *user_dirs* file. Note that **B** will overwrite any existing file. If desired, using *ls -1* will format the output in a single column.

   Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect because *echo* will not list the contents of the folder; you will get a file *users_dirs* that contains the text "/home." While the command in Answer **C** will append the directory listing to the end of the user_dirs file, if the information in the file was captured on an earlier date and that information has changed, you will end up a file that has both current information from the most recent execution of the command and obsolete information from the previous execution. Answer **D** is incorrect because the *cp* command will attempt to copy the /home folder to a new destination. Because there is no destination listed, an error will be generated on the console (because stderr wasn't redirected) and the file *user_dirs* will be created but empty.

2. Correct answers and explanations: **A**. Answer **A** is correct because it will list all running programs, then search for a line that contains "ntp" and show it on the screen. Note that it will only confirm that the process is running, not if it is synchronized.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect because *ps -C ntp* will look for an exact match for "ntp." The actual service is *ntpd*, so no match will be found. Answer **C** is incorrect because *ps* will interpret the *ntp* as an option, which is invalid. Answer **D** is incorrect because while the *ntp* init script supports the *status* flag which will confirm it is running and show the synchronization status; it is located in /etc/ init.d and not /usr/init.d.

3. Correct answer and explanation: **B**. Answer **B** is the correct because the keyword *now* understands adding time using the + sign and descriptive time units. The return is necessary to start the input for what command(s) to run, and the <ctrl>d is needed on a separate line to end the input.

   Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect because the time parameter is wrong, and no units are given. Answer **C** is incorrect because the time format is wrong and the script that is to run is not on a separate line. Answer **D** is incorrect because the time parameter must be given in whole numbers.

4. Correct answer and explanation: **C**. Answer **C** is correct because the first ^ identifies text to be searched for and the second ^ identifies the text that will be inserted in place of the found text.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect because a single ! expects a line number or beginning characters from the history file; it does not take options. The syntax in Answer **B** is incorrect;

the "greater than" character should be a fore slash; the proper syntax is !!s/new-stuff/new_stuff. Answer **D** is incorrect because the history command option *-r* is not used for replacement.

5. Correct answers and explanations: **A**. Answer **A** is correct because it sets the file's permissions to allow read, write and execute for the owner, but only read access for group and everyone else.

Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect because it sets user, group, and everyone else's rights to read and write, but not execute. Answer **C** is incorrect because it adds write permission for the file owner, but not the execute permission, which is the one that is required. Answer **D** is incorrect because it adds the execute permission for the files group. Note that even if you are a member of the files group, if you are the owner you will need user execute rights to execute the file.

CHAPTER 6

Installing Applications

Exam objectives in this chapter

- Installing, Removing, and Updating Programs
- Adding and Removing Repositories

INTRODUCTION

Once Linux is installed and configured, as an administrator most of your time will be spent installing, configuring, supporting, and removing applications. This chapter will walk you through these processes using a variety of methods and tools, with the two most prevalent tools for managing application packages are Advanced Packaging Tool (APT) and Red Hat Package Manager (RPM). When you need to install applications that are not packaged, you will need to resort to compiling and installing from the source code.

INSTALLING, REMOVING, AND UPDATING PROGRAMS

Within the Linux system, there are a number of ways to install programs or *packages,* as they are commonly known within Linux. A package can be considered to be a group of files that are bundled together into one archive file, and the package format you will need depends entirely on the Linux distribution you are running. Whichever package type you need to install, each will usually have one or more associated *libraries* or support packages that have to be installed with it to make it work. These additional packages are called *dependencies*, as the main program is dependent on these to work. If you already have these dependencies installed, then you do not have to reinstall them.

In principal, there are two types of packages: *binary packages* and *source packages.* The source packages need to be compiled and built for your system, whereas the binary packages have already been compiled for a specific installation. The utilities that distribute and manage the binaries for a particular distribution are called *package managers*. Table 6.1 shows the most common ones.

Table 6.1	Main Linux Package Formats
.rpm	RPM Package Manager is used by Red Hat, openSUSE, Mandriva Linux, Mandrake, and many more.
.deb	Debian package is used by Debian and Ubuntu, Knoppix.
tgz or tar.gz	*tar* and *gzip* package is used by Slackware.

The packages can be found individually on various Web sites or on installation disk but also in *software repositories,* which are locations where the software packages can be found and downloaded. These repositories can vary from having a small number of packages (even one) on them or with a whole operating system on them.

Crunch Time

You need to know and understand the following for the exam:

- Packages contain a group of (usually) related program files.
- Packages often have associated library files in them.
- Packages can be binary or source files.
- Dependencies are other packages a program needs installed.
- Software repositories are locations where packages are located.

RPM

RPM stands for RPM Package Manager and is a feature of Red Hat Linux and similar distributions. The basic *rpm* file is a precompiled binary package bundled with a script file, which can be used to build, install, remove, modify, and verify the software archives. The *rpm* packages contain a complete archive of the files, along with a host of other information on the package, such as name, version, and checksums. In addition, there can be scripts included to install, upgrade, or remove the software, along with preinstallation and postinstallation scripts where necessary.

The RPM system uses a database to hold and track all this information, including the version of all the software that is installed. This basic information is held in the /var/lib/rpm directory, and a sample listing of this directory is shown below:

```
$ ls -l
total 46184
-rw-r--r-- 1 root root  2994176 2009-06-23 12:29 Basenames
-rw-r--r-- 1 root root    12288 2009-06-23 12:29 Conflictname
```

```
-rw-r--r-- 1 root root        0 2009-06-23 12:13 __db.000
-rw-r--r-- 1 root root    24576 2009-06-23 12:31 __db.001
-rw-r--r-- 1 root root   180224 2009-06-23 12:31 __db.002
-rw-r--r-- 1 root root  1318912 2009-06-23 12:31 __db.003
-rw-r--r-- 1 root root   352256 2009-06-23 12:31 __db.004
-rw-r--r-- 1 root root  1507328 2009-06-23 12:29 Dirnames
-rw-r--r-- 1 root root  5300224 2009-06-23 12:29 Filedigests
-rw-r--r-- 1 root root    32768 2009-06-23 12:29 Group
-rw-r--r-- 1 root root    24576 2009-06-23 12:29 Installtid
-rw-r--r-- 1 root root    45056 2009-06-23 12:29 Name
-rw-r--r-- 1 root root 35545088 2009-06-23 12:29 Packages
-rw-r--r-- 1 root root   331776 2009-06-23 12:29 Providename
-rw-r--r-- 1 root root   118784 2009-06-23 12:29 Provideversion
-rw-r--r-- 1 root root    12288 2008-11-19 14:17 Pubkeys
-rw-r--r-- 1 root root   503808 2009-06-23 12:29 Requirename
-rw-r--r-- 1 root root   270336 2009-06-23 12:29 Requireversion
-rw-r--r-- 1 root root   163840 2009-06-23 12:29 Sha1header
-rw-r--r-- 1 root root    86016 2009-06-23 12:29 Sigmd5
-rw-r--r-- 1 root root    12288 2009-06-23 12:29 Triggername
```

The /var/lib/rpm/packages file is the primary database of all the installed software in the system and will grow depending on the number of packages you install. A file of 40 MB or larger is not an unusual size for a fully loaded system. This directory is used by RPM to manage all the software and versions.

The *rpm* package names are in a standard format and contain

- Package software name
- Version of the software
- Release number of the package
- Architecture the package was built for (for example, i386, i686).

The actual format will be as below:

```
<package_name>-<version>-<release>.<arch>.rpm
```

As an example, the *rpm* for a Telnet package file, with a version of 0.17 and a release of 23 built for the i386 platform, would look like *telnetd-0.17-23 .i386.rpm*.

COMMAND-LINE TOOLS

Both the graphical user interface (GUI) and the command-line interface are similar in one respect – they need to be executed by the superuser account. The basic format of the *rpm* command is

```
rpm option package_name
```

Fast Facts

The basic options used with the *rpm* command are shown below, and any system administrator needs to learn these:

- *-i* installs the package.
- *-e* removes (erases) the package.
- *-U* removes the installed package first and then installs the new version.
- *-V* verifies the installation of the package.
- *-q* queries the package.

Each of these can be combined with a number of other options to make the *rpm* command very powerful.

Package Installation

Packages are installed using the *–i* option. A simple install of the Telnet package specified above would therefore be

```
rpm -i telnetd-0.17-23.i386.rpm
```

There are number of useful options that are often combined with the install option, as well as with the other options, namely *-v* and *-h*. The *-v* option is for verbose and will give some useful feedback during the process. The *-h* option will print a series of hash marks on the screen as the work proceeds, with the sole purpose of keeping you aware that something is still happening.

Package Updating

Packages already installed on system can be upgraded to a later release using the *–U* option. This option will remove the old version, keeping any modified files such as the configuration files. It will then install the new version. To upgrade the Telnet package above to version 18, release 5, the command to use is

```
rpm -U telnetd-0.18-5.i386.rpm
```

To see this upgrade in verbose mode and printing hash marks when the archive is unpacked, the command would look like

```
rpm -Uvh telnetd-0.18-5.i386.rpm
```

Package Querying

The query command, *-q*, queries the *rpm* database and gives you data on the package. Information on all the packages installed can easily be displayed using *rpm -qa*, but the output will be very long and should be piped through *more* or *less* or redirected to a file.

Package Removing

Packages already installed on a system can be removed or erased with the *-e* option. For instance, removing the Telnet package above can be achieved thus:

```
rpm -e telnetd-0.17-23.i386
```

DID YOU KNOW?

The two most widely used package types within Linux are *rpm* and *deb*.

- Both allow the user to install, update, remove, and query software.
- Additional tools can be used to resolve dependencies (for example, *yum* and *APT*).
- Remove or erase packages with the "e" option with *rpm*.
- Remove packages with the "r" option with *dpkg*.

YUM

There is an automatic installation, removal, and update utility for *rpm* packages called *yellowdog updater modified* (*yum*). The advantage of using *yum* is that it resolves the dependencies automatically. Fedora Linux provides a number of software repositories and is preconfigured to work with three repositories:

- Base repository contains the Fedora release, usually on your installation media.
- Updates have all the updates from the base package.
- Extras is a large selection of additional software that a user may want to install.

In addition, there are also development repositories available which will have the newest code, but which may not be stable. Like *rpm*, *yum* needs to have superuser privileges to be performed.

Installing Software with *yum*

You can install new software packages or package groups with *yum*. The following shows the commands for a single package (firefox) and a package group (MySQL database).

```
su -c 'yum install firefox'

su -c 'yum groupinstall "MySQL Database"'
```

When the *yum* command executes, it checks for any dependencies, resolves them, and then installs the specified package. The dependency check needs to be recursive, so new packages that have to be installed are checked for dependencies and so on.

> **EXAM WARNING**
>
> When you install a service, the Linux system will not start it. You must configure the service to run on bootup, which can be achieved from the command line using *chkconfig* and *service* commands.

Updating Software

To update a software package that is already installed, you can use the *update* option within *yum*. For example, to update the *tsclient* package you would type the command **su –c 'yum update tsclient'**.

If the software being updated is currently in use by the system, the application or service needs to be restarted before the update is made current. The kernel can also be updated using *yum*, and these updates will only come into force upon a restart of the system. When the kernel is updated, *yum* will retain the old version in order that the old kernel can be booted into in case of an error with the new kernel. Only the current and previous versions are kept.

Package groups can also be updated, for example, the MySQL Database package group is updated using the command *su –c 'yum groupinstall "MySQL Database"'*.

Removing Software

The removal of software is achieved using the *remove* option. Again, both packages and package groups can be removed. When this is invoked, *yum* checks the software package and the dependencies and removes both.

```
su -c 'yum remove firefox'

su -c 'yum groupinstall "MySQL Database"'
```

When *yum* removes the software package, it leaves any user data in place, but the configuration files may be removed.

deb

The Debian-derived distributions of Linux are based on the GNU project, and the Debian packages can be considered to be similar to the *rpm* packages in that they are precompiled for easy installation on the target system.

There are three main package libraries available from the Debian package Web site (http://packages.debian.org):

- Stable libraries are well tested and will change only for security or major bug fixes.
- Testing libraries have had a lot of testing and are destined to be in the next release.

- Unstable have had little testing and may well contain bugs that could make the system unstable.

The low level or base tool of the Debian package manager is the command *dpkg*. This command and the main options will be discussed below. In addition to this tool, there are also a number of higher level tools such as *APT*, which can be used to fetch packages from many locations.

INSTALLING SOFTWARE PACKAGES USING dpkg

To install a package, the following is used:

```
dpkg -i package_name
```

As with *rpm* above, you will need to have superuser privileges to run the command. Without this, the following error message will be displayed:

```
dpkg: requested operation requires superuser privilege
```

REMOVING SOFTWARE PACKAGES USING dpkg

Packages can be removed easily using the *–r* option:

```
dpkg -r package_name
```

This option leaves the configuration files on the computer so that it will be easier to reinstall it later. If you want to erase the configuration files as well, you can add the *–purge* option as well.

APT

Another package management tool for Debian is *APT*, which was designed to facilitate the administration of the packages. The default location for the *APT* configuration files is /etc/apt. *APT* has a list of locations where packages can be obtained from, and this is in /etc/apt/sources.list, part of which is shown below:

```
#
# deb cdrom:[Debian GNU/Linux 5.0.1 _Lenny_ - Official i386 DVD Bina-
   ry-1 20090413-00:33]/lenny contrib main

deb cdrom:[Debian GNU/Linux 5.0.1 _Lenny_-Official i386 DVD Binary-1
   20090413-00:33]/lenny contrib main

deb http://security.debian.org/lenny/updates main contrib

deb-src http://security.debian.org/lenny/updates main contrib

deb http://volatile.debian.org/debian-volatile lenny/volatile main
   contrib

deb-src http://volatile.debian.org/debian-volatile lenny/volatile
   main contrib
```

There is an internal database kept by *APT* to track the packages that are currently installed, those that are not installed, and those that are available to

be installed. You can use the *apt-get* command to query this database, install, remove packages, and check for dependencies in packages. As this list changes when new packages are added and new dependencies coming into force, the list needs to be updated. This is achieved using the command:

```
apt-get update
```

INSTALLING PACKAGES

Packages can be installed using the *install* option, with the general syntax of

```
apt-get install package_name(s)
```

An example of the first part of the output is shown below, when the *abiword* package is installed:

```
Reading package lists... Done

Building dependency tree

Reading state information... Done

The following extra packages will be installed:

   abiword-common abiword-help abiword-plugin-grammar
abiword-plugin-mathview

   aspell-en doc-base latex-xft-fonts libaiksaurus-1.2-0c2a

   libaiksaurus-1.2-data libaiksaurusgtk-1.2-0c2a
libfreezethaw-perl

   libfribidi0 libgdome2-0 libgdome2-cpp-smart0c2a
libgoffice-0-4

   libgoffice-0-common libgsf-gnome-1-114 libgtkmathview0c2a
liblink-grammar4

   libloudmouth1-0 libmldbm-perl libots0 libt1-5 libuuid-
perl libwv-1.2-3

   link-grammar-dictionaries-en

Suggested packages:

   abiword-plugin-goffice

The following NEW packages will be installed:

   abiword abiword-common abiword-help abiword-plugin-
grammar

   abiword-plugin-mathview aspell-en doc-base latex-xft-
fonts

   libaiksaurus-1.2-0c2a libaiksaurus-1.2-data
libaiksaurusgtk-1.2-0c2a
```

```
   libfreezethaw-perl libfribidi0 libgdome2-0 libgdome2-cpp-
smart0c2a

   libgoffice-0-4 libgoffice-0-common libgsf-gnome-1-114
libgtkmathview0c2a

   liblink-grammar4 libloudmouth1-0 libmldbm-perl libots0
libt1-5 libuuid-perl

   libwv-1.2-3 link-grammar-dictionaries-en

0 upgraded, 27 newly installed, 0 to remove and 0 not upgraded.

Need to get 0B/9858kB of archives.

After this operation, 31.3MB of additional disk space will
   be used.

Do you want to continue [Y/n]? y

Selecting previously deselected package libaiksaurus-1.2-data.

(Reading database ... 95088 files and directories currently
   installed.)

Unpacking libaiksaurus-1.2-data (from .../libaiksaurus-1.2-
   data_1.2.1+dev-0.12-6_all.deb)...
```

This invokes *apt-get* to search the database for the most recent version of the package and then retrieves it from the location specified in /etc/apt/sources.list, and if there are any dependencies, install these packages as well. The option *–y* can be used with the install option to assume *Yes* to all queries to reduce the user interaction.

PACKAGE REMOVAL

Packages are removed from the system using the *remove* option.

```
apt-get remove package_name(s)
```

The package will be removed, along with all the packages that depend on it. The configuration files will not be removed, but adding the option *–purge* will remove all files associated with the package. Using this option is worthwhile if you know you will not be using this package in the future, as it will clean up the disk and not leave a lot of unwanted files in the filesystem.

UPGRADING PACKAGES

The upgrading of packages can be accomplished very easily within the *APT* system. All the packages within the current distribution can be upgraded with a single command:

```
apt-get upgrade
```

For upgrading the packages to a new distribution, it is better to use the command below to ensure that all relationships between packages are updated:

```
apt-get dist-upgrade
```

For both commands, it is worth adding the option *–u* to ensure that there is sufficient output for you to see what is being upgraded. The initial part of the output is shown below:

```
Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages will be upgraded:

  libcupsimage2 libcupsys2 libdns45 libebook1.2-9
libecal1.2-7

  libedata-book1.2-2 libedata-cal1.2-6 libedataserver1.2-9

  libedataserverui1.2-8 libegroupwise1.2-13 libexchange-
storage1.2-3

  libfreetype6 libgdata-google1.2-1 libgdata1.2-1
libglib2.0-0 libicu38

  libisc45 libisccc40 libisccfg40 libkrb53 liblwres40
libmozjs1d

  libmysqlclient15off libnss3-1d libpango1.0-0 libpango1.0-
common

  libpoppler-glib3 libpoppler3 libpostproc51 libpurple0
libsasl2-2

  libsasl2-modules libsmbclient libssl0.9.8 libvolume-id0
libwbclient0
```

OBTAINING INFORMATION ABOUT PACKAGES

You may often want to install a package but not sure what the name of the package is. One method of finding these packages is to use *apt-cache*. To search for packages, you will need to use:

```
apt-cache search name
```

For instance, suppose you want to search for *abiword*, you would execute the command *apt-cache search abiword*, with the output shown below:

```
abiword - efficient, featureful word processor with
  collaboration

abiword-common - efficient, featureful word processor with
  collaboration -- common files
```

```
abiword-help - online help for AbiWord
libgtkmathview0c2a - rendering engine for MathML documents
abiword-plugin-grammar - grammar checking plugin for AbiWord
abiword-plugin-mathview - equation editor plugin for AbiWord
abiword-plugin-goffice - GOffice interaction plugin for AbiWord
```

Resolving Application Dependencies

You can download dependent application packages manually as you try to install an application from an .rpm package, or you can use *Yum*. *Yum* is used on RPM-based systems, such as Red Hat and Fedora. For Linux distributions that use *dpkg*, *APT* natively resolves application dependencies. Although *APT* is a terrific tool from the command line, you are not forced to open a terminal window or change runlevels to drop to a command line when working in a GUI environment.

Compiling and Installing Applications from Source

Compiling and installing from source presents a terrific opportunity to tune applications to your specific hardware and software platform. This section starts with a description of how and where to include these hardware- and software-specific parameters and then continues through the process to compile and install the application. It concludes with several prominent utilities for archiving and packaging source code: *tar*, *bzip*, and *gzip*.

CONFIGURING THE SOURCE

We will discuss downloading the package archive later on in the section, but suppose for now that you have the source in a suitable directory on your hard drive. The first place to look is to see if there are any readme or install files in the directory and read them. These usually contain very useful information on the software and often will give details on how to install them with any specific options or dependencies that you may need.

After reading the documentation, you need to change directories to the directory where the package is stored. You can then configure the package, which is usually achieved using the configure script by typing **./configure**.

The configure script is a shell script which configures the *makefile*, which is used by the compilation tool *make* (described below). This *makefile* will have information on your system to enable *make* to compile the source correctly. The machine output from this command will be a new *makefile*, and if the command worked correctly, this will be constructed and placed in the correct directory. There could be a lot of messages scrolled to the standard output during this process. If it finds an error, it will be reported, and *configure* will exit. If there are no errors, *configure* will end gracefully.

MAKE

The utility *make* is used to automatically determine which components of a software package need to be complied and then to guide this compile process. *make* uses the *makefile,* which details the relationships among the files in the package and how to update these files. This will be undertaken from the data in the database and the last modification times of the files. The executables are typically made up from object files, which themselves are compiled from source files.

Once you have the *makefile* after running compile, then you can run *make*. This shell script is typically run initially with no options and will parse the *makefile* and update any files as necessary. If *make* completes, this will build a binary of the software package. This does not install the binary; that step is achieved in the last step using the command *make install* and will have to be run with superuser privileges.

If this exits with no errors displayed, then the software has been installed correctly. The configure script determines where the program will be installed, typically in /usr/local/bin. To clean up your system from the temporary files left by *make*, you can run the command *make clean*.

AUTOCONF

The *autoconf* utility is a package of M4 macros that are used to build a set of shell scripts to automatically configure software source packages. The utility will create a configuration script from a template file listing the operating system features that the package uses. The generation of the configuration files is primarily to make the user's experience easier, to ensure that the configure process is easier to use and less prone to errors.

Archive Files

The *tar* file format has been in existence since the early days of UNIX and was originally designed for tape archives (*tar*). The utility to use this file format is also called *tar* and is now a common method for archiving or collecting a large number of files into one larger file, while preserving all the file information, such as directory structures, dates, and user and group permissions. Files that are packed into this format have a naming structure of *filename.tar*. These large files can be compressed using a compression utility such as *gzip*, *bzip*, or compress. Depending on the compression utility used, the *tar* file will be renamed.

- filename.tar becomes filename.tar.gz if *gzip* is used.
- filename.tar becomes filename.tar.bz or filename.tar.bz 2 if *bzip/bzip2* is used.

To create a *tar* file, the following general syntax can be used:

```
tar -cvf filename.tar files|directories
```

The options used are *c* to create a *tar* file, *v* for verbose output, and *f* to put the output in the specified file. The *tar* archive will be created from one or more files and/or directories specified at the end. There could be multiple files and directories specified on the same command line. For instance, suppose you wanted to compress everything in your work directory in your home folder, say /home/syngress/work. The command to create an archive work.tar in the current directory would be

```
tar -cvf work.tar /home/syngress/work
```

Once the *tar* file has been created, you can list its contents by using *tar –tvf work.tar*. The *tar* file can be decompressed or the files extracted using *tar –xvf work.tar*. This extraction process does not remove the *tar* file but places the files in the current working directory.

The *tar* utility can also be used to compress the *tar* file that has been created. In the above example, to compress the *tar* file of the work directory, you would use:

```
tar -czvf work.tar
```

The files are compressed using *gzip* and will be given the .tgz extension. The compressed file can be decompressed using

```
tar -xzvf work.tar
```

COMPRESSION UTILITIES

Some compression utilities work best on certain types of file, but we will concentrate on a couple of them. We have just mentioned *gzip* that can be used with *tar*. It is also a stand-alone program that can be invoked to compress files at any time. The format of the command is

```
gzip filename.ext
```

This will compress filename.ext and save it as filename.ext.gz. The original file will be deleted during the process. This can be decompressed by using the command:

```
gunzip filename.ext.gz
```

Again, the command will delete filename.ext.gz and leave filename.ext only. The *gzip* utility can compress files to different levels from 1 to 9, with 1 being quick but least efficient to 9 being slow but very efficient with the default is a level 6. For both *gzip* and *gunzip*, the option *–r* can be used, which will recursively compress or decompress all the files in the current directory and the subdirectories.

The *bzip2/bunzip2* utilities are another pair of compression and decompression tools, which often give slightly better compression ratios. The command-line options are very similar to that of *gzip/gunzip*. The compressed files will usually have an extension of bz/bz2 or tbz/tbz2 (compressed file or a compressed *tar* file).

ADDING AND REMOVING REPOSITORIES

There are thousands of software packages for Linux stored in software repositories, and the main repositories are usually set up at install time. If you are not connected to the Internet during the initial load of Linux, these may not be set up or may be marked as inactive.

yum Repositories

Software repositories can be defined on remote servers as well as locally. The repositories are defined in the /etc/yum.conf file and in /etc/yum.repos.d directory. You can make a local repository by downloading the software from other repositories and then setting up a local repository to save downloading these for a number of machines on your network. When you have downloaded the packages, you need to generate the correct information for a repository. This is achieved using *createrepo*, which extracts all the data from the *rpm* files to generate the necessary metadata for *yum*. The command to create the metadata from the *rpm* files in the /rpm_directory is

```
createrepo /rpm_directory
```

This can then be included into the file in the /etc/yum.repo.d directory.

Fast Facts

Depending on your environment, you may use different repositories depending on the actual use of the system.

- For servers and general users, repositories for the stable distribution are recommended.
- If you need the additional functionality of new, development, or unstable repositories, allow adequate testing time before deployment.
- Users who can add packages from any repository may make their system unstable and add work for your system administrators.

Adding a Repository in Debian

The method of adding a repository that use the *APT* packaging tool is different. The file /etc/apt/sources.list contains a list of available software repositories, and it can be updated manually or (if installed) by a graphical manager tool. If you want to add a new repository manually, the format to follow is package type, Web address (URL), distribution, and section. For example, one of the lines in sources.list could be

```
deb http://security.debian.org/ lenny/updates main contrib
```

SUMMARY OF EXAM OBJECTIVES

In this chapter, you learned how to download and install applications using a variety of methods, both from binary packages and using source code. Initially, we looked at the software package formats that you will most likely encounter on the Linux systems you are administrating – RPM and DEB packages. With both the software package formats, you were guided through the main aspects of software management, namely adding, deleting, and updating the various packages. Although both command sets are very similar to each other, there are a number of differences which you should understand and remember for the exam. The update commands for both should be memorized, and any additional options that can be added to these commands understood.

The downloading, compiling, and building software from source was described, and some background on when and why this may be necessary. The instructions for undertaking this and any command-line switches that may be necessary are usually found in the install text file bundled with the software.

Linux packages often have dependencies when the installation of one package will depend on another package to be installed. This can be resolved using tools such as *yum* and *apt* that will work out what are all the dependencies and install these as well.

All these packages are located in software repositories, which can be considered to be buckets containing one or more software packages. Once these are defined on your system, when you want to download a new software package, the system will look into these repositories to download the package and install it. In addition, updates to the system will be located in the software repositories, and the system will use these to compare with the version of software you have loaded and suggest that any are upgraded if a newer version is released.

TOP FIVE TOUGHEST QUESTIONS

1. You have downloaded the *rpm* files for a new program. Assuming that you are a normal user, what else must you do to ensure that you can install the programs?
   - **A.** Change the owner of the files to everyone, and run *rpm*.
   - **B.** Run chmod 777 on the files before executing the *rpm* command.
   - **C.** Use the command *rpm –c rpmfile* to ensure that the system prompt you for the superuser password.
   - **D.** Run *rpm* as superuser.

2. You want to set up a local repository on a server in your network. You are using *yum* and want to ensure that the repositories will work with this. What tool should you use, and where should the metadata files be stored?
   - **A.** Use *createrepo* and store the metadata in /etc/yum.repo.d.
   - **B.** Use *create-repo* and store the metadata in /etc/yum.repo.d.

C. Use *createrepo* and store the metadata in /etc/yum/yum.repo.d.
D. Use *createrepo* and the metadata will be stored automatically in the correct location.

3. Which of the following commands will not upgrade an installed .deb package?
   A. *apt-get install package_name*
   B. *dpkg -i package_name*
   C. *apt-get --reinstall install package_name*
   D. *apt-get update package_name*

4. You are compiling source code for installation, and you want to string all the required commands together to run while you are going downstairs to grab a coffee so that the binary file is ready when you return. What answer below has syntax that will work?
   A. ./configure; make; make install
   B. ./configure / make / make install
   C. ./configure & make & make install
   D. ./configure | make | make install

5. You are powering up a laptop that has Linux installed and that has not been used in a couple of months. Because you will be handing it over to a user who needs to use it on a long trip, you want to ensure that its applications are current. What command do you run once the *APT* database is up-to-date?
   A. *apt-get update*
   B. *apt-get upgrade*
   C. *apt-get dist-upgrade*
   D. *apt-get install*

ANSWERS

1. Correct answer and explanation: **D**. Answer **D** is correct, as *rpm* must be run with superuser privileges.

   Incorrect answers and explanations: **A**, **B**, and **C**. Answer **A** is incorrect, as you do not need to change the ownership of the files and you need to have superuser privileges. Answer **B** is incorrect, as you do not need to change the permissions of the files and you need to have superuser privileges. Answer **C** is incorrect, as the *–c* option will not make the system prompt you for a superuser password.

2. Correct answer and explanation: **A**. Answer **A** is correct, as you need to use *createrepo* and then store this data in its own file in /etc/yum.repo.d.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect, as there is no *create-repo* command. Answer **C** is incorrect, as the correct file location is normally /etc/yum.repo.d. Answer **D** is incorrect, as *createrepo* will not store the data automatically.

3. Correct answer and explanation: **D**. Answer **D** is correct because *apt-get update* performs an update of the *APT* database but does not upgrade any of the installed applications themselves. Furthermore, the syntax is incorrect because a package name is not used with *apt-get update*.

Incorrect answers and explanations: **A**, **B**, and **C**. Answers **A**, **B**, and **C** are incorrect because all three these commands can be used to install or upgrade an installed .deb package.

4. Correct answer and explanation: **A**. Answer **A** is correct because semicolon command allows for commands to be concatenated, and hence this will work.

Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect because the forward slash is not a valid character for use in concatenating commands. It is used in denoting filesystem locations, such as the root directory, "/". Answer **C** is incorrect because double ampersands not a single ampersand allow for concatenation of commands, and this allows all the commands to execute one after another. Answer **D** is incorrect, as the pipe command "|" allows for the output of the first command to be input into the second command, but the string above will result in an error not a complied program.

5. Correct answer and explanation: **B**. Answer **B** is correct because *apt-get upgrade* uses the information in the *APT* database (which is updated using *apt-get update*) to update all installed applications to their most current versions.

Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect because it is the command to update the *APT* database. Answer **C** is incorrect because the *dist-upgrade* option upgrades the entire distribution to the latest version, and the requirements in this question ask for only the installed applications to be updated. Answer **D** is incorrect because when a package name is specified, it is the command to install an individual application.

CHAPTER 7

Installing, Configuring as a Workstation

Exam objectives in this chapter

- Printing
- X11

INTRODUCTION

Users of any system will want to undertake a number of basic tasks: interaction with applications on a monitor (for example, word processing, task scheduling) and printing. A standard setup for the user interface and the printers will make the users experience much more fulfilling; as well as reduce the support overhead. The section on printers will describe how Common UNIX Printing System (CUPS) is configured and used.

Most of the latest Linux distributions offer one or more graphical user interfaces (GUIs) to interact with the user. These are commonly run on top of the X Windows system, with the most common window managers being KDE and GNOME. X11, the latest incarnation of X Windows is a true client/server application, and how this is implemented is described, along with descriptions of the main configuration files that are needed.

PRINTING

The management of the devices and the setup can be confusing as the support for Linux print drivers varies between manufacturers, and not all have compatible Linux drivers. CUPS is a print service program that is commonly deployed in a modern Linux distribution.

CUPS Overview

The CUPS application will convert the page descriptions from an application into a format that the printer will understand and then manages the process for

sending this to the actual printer. Manufactures of printers will develop different methods to print, even within their own line of printers. The CUPS application will perform this conversion, hopefully hidden from the end user.

Fast Facts

The CUPS application can be summarized as performing the following actions.

- Create and manage a queue for each printer.
- Users will send print requests to CUPS and CUPS puts the job into the appropriate queue.
- Each job will have a job request associated with it.
- Users can cancel or pause their own jobs.
- CUPS assigns a program to each job to best convert it into a printable form.
- Jobs are actioned in a first-in/first-out basis.
- Completed jobs are removed from the queue by CUPS.

Enable and Disable Queues

Each printer will have its own queue which will be managed to ensure that the jobs are printed in an orderly fashion. These queues can be enabled or disabled, even if there are items in the queue. This is often necessary if a printer has a problem that needs to be fixed and the system administrator wants to disable the queue until the issue is corrected. A system administrator can also issue a comment to tell users why the printer is disabled.

```
# cupsdisable -r "Printer maintenance being performed" CANONMX
```

The printer queue can be enabled using the command:

```
# cupsenable CANONMX
```

WEB MANAGEMENT PORT (PORT 631)

The interface to CUPS is via a Web interface that allows you to view print jobs and what printers are installed, and will also to manage of them. CUPS can be accessed via a Web interface on port 631, either locally or remotely. On the local machine, the interface can be accessed using the URL *http://localhost:631*. The interface is shown in Figure 7.1. In addition, there is a CUPS application program available, which does not use a browser, but the look and feel is identical.

There are some pages on the GUI that will require a user name and password to perform some actions on them, such as to add a printer. On most Linux distributions, only root can add, modify, or delete a printer or class of printers.

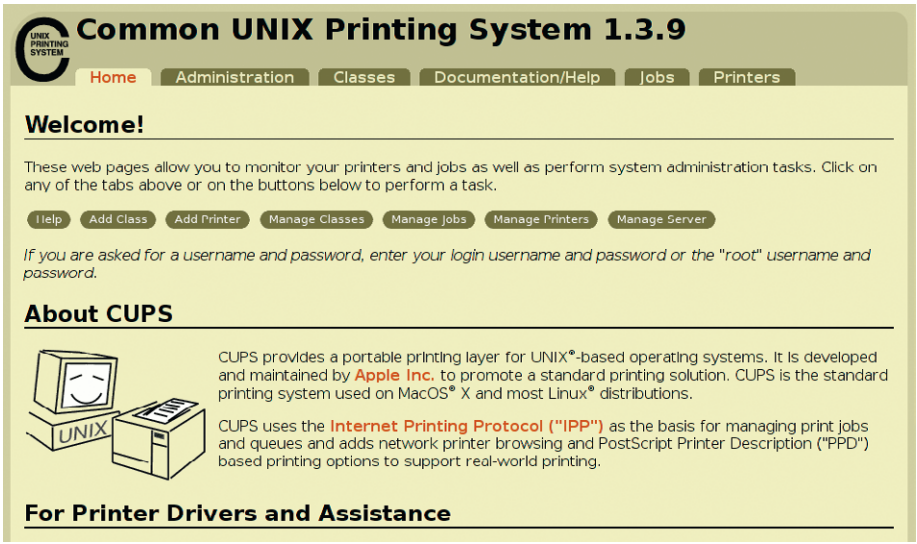

FIGURE 7.1
CUPS Interface in a Browser

Crunch Time

You must remember the port numbers and names that are commonly used as references to the port name or number may be used. The Web interface port 631 is one of the required ports to remember.

Managing Printers, Jobs, and Queues

The CUPS management interface can be used to add and delete printers, and this is accessed under the *Administration* tab. Local printers can be added directly if you know the make and model or the system can find new printers that have been attached to it.

Printer jobs can be managed via two options: directly from the first page in the administration tab or by clicking on "manage printers" under the administration tab and managing the jobs from there.

Fast Facts

The management of printers section is very powerful and allows the user to

- Print a test page
- Stop/start a printer
- Enable and disable a print queue

- Move jobs from one print queue to another
- Cancel all jobs
- Set printer options
- Allow specified users
- Set as default printer

Later on in this chapter, in the Printing Commands section, we will look at the command line versions of some of these commands. The CUPS interface is very easy to understand and for the majority of users, it is easier to use than the command line. As the system administrator can prevent unauthorized users changing certain functions, it provides an easy, safe GUI to deploy to users.

CUPS Printer Classes

CUPS allows the user to group printers together and this collection of printers is called a *class*. This is particularly of interest in a company who has a number of printers and they can be grouped together in areas, such as finance, admin, and so forth. This allows printers to be taken offline for maintenance and the user will not notice any disruption or need to remember the names of different printers in their area. The adding and management of printer classes is undertaken in the Administration tab of the CUPS GUI.

DID YOU KNOW?

Although we have discussed CUPS in terms of Linux, it is a cross-platform software application.

- CUPS can run on Microsoft Windows and Apple platforms as well.
- The application can manage local and remote printers – even across the Internet.
- The Web-based GUI allows remote administration.
- The Web interface port is 631.

Printing Commands

Once a printer has been installed and properly configured, a user is able to print to it from any printer-capable graphical client. Text-based interface can also print via a simple command line interface. These basic commands are described in the following sections.

LPR

The *lpr* command submits print jobs to the specified printer, or the default printer if none is specified. The main options that a user will need are shown in Table 7.1.

Table 7.1 *lpr* Printing Options

-P destination	The name of the printer to send the job to.
-# number	Print a specific number of copies (default is 1 copy).
-T title	Prints a title on the banner page of the output.
-h	Suppresses printing of the banner page.
-w cols	Prints file with pages of a specific width.
-m	Send mail when the job has printed.
Filename	The name of the file which you want to print.

This command is used with BSD systems.

LP

The *lp* command is very similar to the *lpr* command described earlier, but it works on System V systems. There are a number of differences in the syntax, such as the –o parameter with *lp* has a number of different options such as *nobanner*, *cpi=pitch*, and *width=chars*. Depending on which system you are working on, you should understand the syntax of each command.

LPQ

lpq is used to see the status of one or more print queues and this can be run once or displayed continuously at a specified interval until the queue is empty. The command on its own will display the queue of the default printer. The main options for the command are as follows:

-a to show the queue status of all printers
-P status of a specific printer
+interval will display the queue every *interval* seconds until empty

LPSTAT

The *lpstat* command can show the status of printers and queues like the *lpq* command; however, there are far more options, such as options to query the status of specific CUPS servers in the network. The options are shown in Table 7.2. This is useful for system administrators who can administer print servers and queues from a central location.

CANCEL

Administrators can cancel jobs still in the queue of a specific printer. An individual job can be canceled or all jobs can be removed, if the user has the appropriate rights. The default printer queue will be checked if no specific printer is defined.

Table 7.2 *lpstat* Options

-a	Display the queues for all printers
-d	Display the default destination
-h *server*	Specifies the *CUPS server* to communicate to
-o *printer*	Display the queue on printer
-p *printer*	Shows status of printer
-r	Status of CUPS server

X11

X11 is the common name for the X Windows system, which is the GUI found on most Linux distributions today.

Starting and Stopping X11

There are a number of ways to start and stop the X Windows system. Normally, the display manager will be loaded automatically on boot, accomplished by setting up the /etc/inittab file to load the X windows system when multiuser mode is used (runlevel 5). The actual display manager that will be used is defined in the /etc/sysinit/displaymanager configuration file as shown below for the KDE environment:

```
DISPLAYMANAGER="kdm4"
```

When Linux is booted into a multiuser mode without a graphical login using the display manager, such as runlevel 3 in the /etc/inittab file, you will have to start an X session from the command line. The X server and X session must be started, and this can be accomplished using *startx*, often without any parameters.

The *startx* command will use configuration settings found in the .xinitrc file which by default will be in /etc/X11/xinit directory; however, the user can customize the X session and launch default clients using a .xinitrc file located in their home directory. If you intend to use the display manager that was installed with your system, you do not need to use a customized .xinitrc file.

The *startx* command itself can be passed parameters which it passes to the X server before an X session is launched. One such parameter is the *–depth* option that will alter the number of colors used. This can be used during development or testing of X clients to ascertain how each will look on different hardware solutions. This may be useful if you are planning a large roll out of Linux across a variety of differing hardware. This will start an X session a depth of 16.7 million colors.

```
$ startx -- -depth 24
```

The *startx* command can also be used to launch multiple X sessions in a number of virtual consoles. The complexity of multiple X sessions means that it is not an everyday occurrence, and you may wish to be very familiar with X sessions

before you try this. Each X session then to be closed separately or the system gets restarted in runlevel 3 as mentioned earlier.

Difference between X11 Clients and Server

The X Window System was developed using a client/server model, which means that the two components can be distributed on separate systems. On a typical Linux configuration, both these components are located on the same machine; however, they communicate via the standard processes. In a client/server application, the client software process will initiate a communication session, while the server waits for a request from one of its clients. Most people use client/server programs on a regular basis without realizing it. For instance, a Web browser is a client program that requests data from a Web server.

In X11, the X server communicates with various client programs. This server accepts requests for graphical output, that is, the normal display window on a screen and also accepts user input from the keyboard and mouse and sends these back to the client. The server can send out to a display on another system or may control the video output on the local system. Thus, the user's terminal is the server and the applications are the client, which is often the cause of confusion for new users as they typically think of a server from their perspective as an end-user. Their normal perception is that the client runs on the user's computer and the server is remote. In X Window terminology, the server provides display and I/O services to the applications. The applications that use these services are clients (such as a browser).

The Server and Client Communication Protocol runs with network transparency, that is, it is invisible to the applications that are using it. When Linux is designated as a server, the clients are often distributed across the network. This Communication Protocol can be tunneled over an encrypted tunnel. X can be bandwidth intensive, so a fast network is desired, particularly if there a lot of remote clients.

Window Managers

The X Window manager is the windowing system that runs on X, and the user can choose one of many. These windows manager will have certain requests between the client and server redirected through them, such as when a new window needs to be displayed. The types of window manager include tiling window managers (such as ion, dwm, and so forth), compositing window managers (GNOME and KDE are good common examples), and stacking window managers, such as IceWM.

In general, the window manager controls the appearance of the GUI to the user. It positions the windows, controls the fonts and colors, and handles the input and output (mouse-clicks and so forth). The window manager is just another client from the point of view of the X window server. The initial window that is displayed is defined as the root window, with top-level windows being children of this root window.

X SESSION MANAGER

The state of the desktop or session at a given time is managed by the X Session Manager. This allows, for instance, a user to log out and then log into a session and to have the same windows displayed. The session manager stores the state of all the windows on exit to restore them. Although the default session manager *xsm* can be used, specific session managers are often bundled with the display manager, such as *ksmserver* in KDE.

DISPLAY MANAGER

The X display manager runs as a program that allows the X server to start a session on a system, either local to the server or remotely. The display manager often prompts the user for a user name and a password with an initial login screen, although it may be bypassed depending on the set up of the system (usually via KDM's auto login feature). When it is run on a local system, it will start the X server before presenting the login screen. In a remote session, the display manager will act like a telnet server requesting a username and password and starting the remote session. The default display manager in X is the X Windows Display Manager (XDM). The most popular alternative display managers are explained later. Alternative display managers are often used, as the default XDM is often more complex to configure for the standard user.

KDM

The display manager bundled with KDE is the KDE Display Manager, KDM. It is very configurable from the KDE control center, allows configuring screen color, menu options, styles, and so on. KDM was originally based on XDM. The main configuration file for KDM is often found in */usr/share/kde4/config/kdm* and is called *kdmrc*. KDM must be run before a user is logged in, so it is therefore not associated with any user and hence user-specific configuration files are not possible. Users can, however, change the appearance of their desktop once they are logged in.

The graphical login manager within KDM is *The Greeting*, which can show a company logo, current system time, or perhaps nothing. Some or all users may be allowed to shutdown the system from this initial screen.

GDM

GNOME was conceived slightly later than KDE and unlike KM, this was written entirely from scratch and does not contain any original XDM code. Upon startup, the GDM daemon reads its local configuration file, *gdm.conf*. When there is more than one local display, each of these forks an Xserver and slave process. The initial process will start the display and calls *gdmlogin* to prompt the user for a user name and a password. Remote displays are managed using the X Display Manager Protocol, *XDMCP*, typically running on port 177.

The configuration file can be found in the file */etc/gdm/gdm.conf* and the syntax within the file follows the standard GNOME file syntax. The file contains a large number of options for the daemon configuration, security, and remote and local GUI. When the system is installed, most of these are set up automatically. The

look and feel of GDM can be configured by a large number of themes that are available in the main package or downloadable from various sites.

Differences between KDM and GDM

At first glance, the differences between GDM and KDM are purely cosmetic (colors, whether there is a toolbar at the bottom of the screen or main menu, and so on). Overall, it is often said that a system installed with KDE is more Microsoft Windows like than one with GNOME.

GDM by default has two toolbars at the top and bottom, and it also divides its menu into three submenus: *applications*, *places*, and *system*. KDM has only one toolbar at the bottom and its menu is divided into *favorites*, *applications*, and *computer*. In terms of user input, the default for KDM is one click, and two in GDM. Also, the system configuration menus are usually more complicated in KDM as opposed to GDM.

You can install both window managers onto a system and switch between them (or have them display on different monitors at the same time, if you have enough memory and computing power). This is undertaken in many ways, with some common distributions like Fedora using the *switchdesk* command.

Multiple Desktops

Within the X Windows systems, there is an ability to have more than one desktop. Each desktop is known as a virtual desktop within the system, and each can operate independently to one another. This concept allows the user to create two or more separate environments where simultaneous tasks can be undertaken. This may involve a word processor in one desktop, email client open in another, and perhaps some scripts running in a terminal session in another. The use of virtual desktops allows an uncluttered view of the applications with the ability to change to another desktop at the click of a button.

KDE VIRTUAL DESKTOP

There are four virtual desktops installed with KDE as a default, with the option to install a total of 36. These are shown on the bottom toolbar, initially numbered 1 to 4.

Moving between desktops is achieved just by clicking on the appropriate number. The standard desktop is initially installed on all desktops. To make it easier for navigation, these can be renamed using the multiple desktop control module as shown in Figure 7.2.

The toolbar will then appear as shown in Figure 7.3.

GNOME WORKSPACES

Within the GNOME system, these are called workspaces and there is a *workspace switcher application* that controls this. You can set different preferences including the number of workspaces you want (up to 36). The user can move to a different workspace using the applet or you can move to one by pressing **Ctrl**

and scrolling the mouse to highlight the desired workspace. A part of a typical GNOME screen showing four numbered workspaces is shown in Figure 7.4. Rather than having names such as Desk 1, Desk 2, and so on, you can rename the workplaces to more meaningful names, perhaps mail, scripts, Internet, and so forth. Individual applications within one desktop can be moved to another desktop or made to be visible in all workspaces.

X Window System Directories

The current implementation of X controlled by the X.org foundation (www.x.org) is a distribution commonly called *Xorg*, typically more than 100 MB in size.

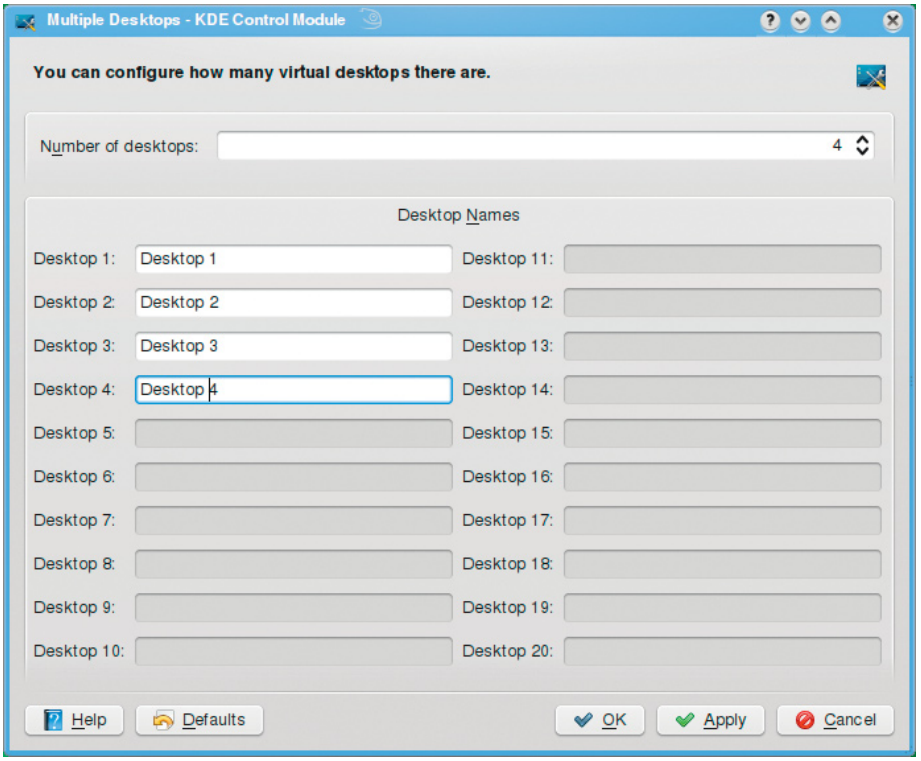

FIGURE 7.2
Multiple Desktop Control Module

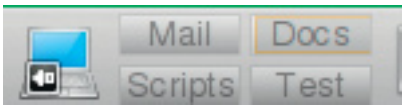

FIGURE 7.3
KDE Toolbar Showing Named Virtual Desktops

FIGURE 7.4
Default GNOME Workspace

Fast Facts

The /usr directory and its subdirectories contain the majority of Xorg's software, with the most significant directories being:

- */usr/bin*: It contains the X server and the clients that are installed.
- */usr/lib*: It contains software libraries in support of the X server.
- */usr/include*: It has the include files needed if a new X client needs to be developed.
- */usr/lib/x11*: A symbolic link will be found to this directory in /usr/lib. It contains the font file, documentation, system, and client resources.
- /usr/lib/modules: This will hold all the drivers for various graphics cards used by the X server.
- /usr/X11/man: The man pages for X11.

When Linux is installed on a computer and the graphical desktop option installed, the main components required to run a local X session are loaded: X server, basic fonts, a terminal client, and a Window Manager. The main configuration file for the X Windows system is the *xorg.conf* file. The xorg.conf file is typically located in /etc/X11/xorg.conf, although this does vary across some Linux distributions.

The xorg.conf file will contain a number of different sections that will provide information on the input devices, monitor layout, file locations, and so on. The main sections are as follows:

- *Files*: The location of the fonts and colors of the X server.
- *Module*: It informs the X server on the dynamic modules to load.
- *InputDevice*: It defines all the input devices such as the mouse and keyboard that are used in the system.
- *ServerLayout*: It defines the display and defines the screen layouts.
- *Monitor*: Attributes of the monitor(s) that are attached to the system.
- *Screen*: Screen configuration.
- *Device*: It defines all the graphics cards attached to the system.

The ServerLayout sections are at the highest level and can bind together the input and output devices to be used in a session. The output devices will comprise of a number of sections (for example the graphics board and a monitor). All the input and output devices are bound together in the Screen section and these are used in the ServerLayout section. Although most of the settings for the input and output sections are configured automatically, some users may wish to override these, for example to force the monitor to a lower resolution.

EXAM WARNING

The name and location of the main configuration files are important to know for the exam. In particular, make sure that you know the configuration files you need to change to configure individual users desktop.

Terminal Emulators

Within Linux, there are a number of built-in terminal emulators, often bundled with the X Windows display manager. The two common ones are *gnome-terminal* and *kconsole* in GNOME and KDE, respectively. These are feature rich and have a very user-friendly interface, although both have a relatively high memory foot print. For everyday tasks where memory is not an issue, these terminal emulators are very good and easy to use and are probably the preferred interface for a user. However, for systems that have a smaller amount of RAM, there is another emulator called *xterm* that has a very small memory footprint (usually under 1 MB), although the windows is smaller and the default font is hard to read. The standard xterm emulations are for DEC VTxxx and Tektronix 4014 terminals.

The VTxxx and Tektronix terminals can each have their own window so that a user can edit text in one and look at graphics in another. Only one of these windows will be considered active at any one time, and this one will accept keyboard input and terminal output. The number of options allowed for xterm is very large. That will allow the system to correctly emulate the terminal and application running in it.

SUMMARY OF EXAM OBJECTIVES

In this chapter, you learned about how to configure a Linux system when it is used as a workstation. The two most common aspects of the user experience, namely printing and the GUI (X Windows interface), and how these can be configured for individual users were explained.

The CUPS interface was shown, and how to add local printers using the interface was explained. The management of printers and their queues was demonstrated using the CUPS interface and also how this could be achieved using the command line interface.

The user's main interface to the computer is through the GUI, and the number of different interfaces that can be used is very large. The standard, underlying interface is X Windows and is commonly known as X11. This can be used and enhanced with many different display managers; the two most common ones being GNOME and KDE. The main configuration files for X Windows and how to modify these for individual users were described. Specific configuration details were then outlined for the GNOME and KDE desktops. The use of multiple or virtual desktops and how these may be different from a user's experience of a single desktop were explained. The configuration of these multiple desktops was explained.

TOP FIVE TOUGHEST QUESTIONS

1. A user has problems with the startup of their system and the user wishes to start up their system in single-user mode and then to start X Windows. Which is the best method to achieve this?
   - **A.** Start the system in runlevel 1 and then run *startx*.
   - **B.** Start the system in runlevel 5 and then run *startx*.
   - **C.** Reboot the system and when the initial load screen appears, press **Ctrl + S** together.
   - **D.** Reboot the system and when the initial load screen appears, press **Ctrl + 1** together.

2. You have just downloaded and installed the latest version of the GNOME desktop. This is a beta version and your system seems to freeze when you start it. Which option would be the best one to use?
   - **A.** Press the reset button and boot into single-user mode with command line input. Then remove the beta version.
   - **B.** Open a terminal window and type **shutdown –now** and then boot into single-user mode with command line input. Then remove the beta version.
   - **C.** Press **Ctrl + Alt + F2** and use the root username and password when prompted. Look at the PID list and kill all the processes associated with X Terminal session. You can now uninstall the beta version.
   - **D.** Press **Ctrl + Alt + F7** and use the root username and password when prompted. Type **rollback X11** to revert to the previous version of X Windows.

3. You want to run the GNOME window manager if you boot your system into runlevel 5, and the Openbox window manager if you start the system in runlevel 4. How can this be best achieved?
   - **A.** You cannot start a different window manager based on runlevel.
   - **B.** Modify the .xinitrc in your home directory and include shell code to execute the commands *exec gnome-session* or *exec openbox-session* based on the current runlevel.
   - **C.** Boot the system directly into a terminal session and run a script to start X based on the appropriate runlevel.
   - **D.** Add the lines in the .xinitrc file in your home directory to switch window managers:

```
if runlevel=4 then
gnome-session else
openbox-session
end
```

4. You are a system administrator for a large company and a user wants to purchase a new color printer for his administrative assistant to produce sales literature. This printer is not the usual printer you purchase. What

would be your best advice to the user to ensure that the new printer works with their Linux system?

A. Find out the make and models of the printers he is considering to purchase and check with the www.cups.org Web site to see if the printer's drivers can be downloaded.

B. Look at the printer manufacturer's Web site to see if the printers are listed as "Plug and Play" devices and hence will work seamlessly.

C. Tell the user that he can buy any printer which has the "CUPS Compatible" logo on the box.

D. Tell the user that he can buy any HP printer as they are all compatible with Linux via downloads on HP's Web site.

5. You have sent a job to your default printer and have seen that there are a lot of jobs before it. As you need the printout in a hurry, which is the best option?

A. Using the CUPS GUI finds out the use of an idle printer and moves your job to that printer.

B. Move your job to the top of the queue on your default printer using the CUPS GUI.

C. Login as superuser on the CUPS GUI. Pause all the jobs on the printer ahead of your job so your job will start next

D. Resend your job to the printer using the option *–priority* on the *lpr* command that will insert the job to the head of the queue.

ANSWERS

1. Correct answer and explanation: **A**. Answer **A** is correct as this will start Linux in single-user mode on most Linux distributions and will then allow the user to start X Windows when the command prompt appears.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect, as runlevel 5 normally starts X Windows at the end of the boot process. Answers **C** and **D** are incorrect, as these commands are not available.

2. Correct answer and explanation: **C**. Answer **C** is the correct. This will revert you to a terminal window where you can gracefully kill the rogue X Windows sessions and then remove the beta software.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect, as this will result in the loss of files that have not been written to the disk, and possible disk corruption. It should only be used as a last resort. Answer **B** is also incorrect, as this will also shut the system down immediately with the same consequences as **A**. Answer **D** is incorrect, as there is no *rollback* command to revert the software back to a previous version.

3. Correct answer and explanation: **B**. Answer **B** is correct, as you can test for the appropriate runlevel in this shell script and call the appropriate window manager using the *exec* command.

Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect, as you can start different window managers. Answer **C** could be made to work, but this is not the best way to achieve the result as it is not automatic. Answer **D** is incorrect, as the code is wrong and there is no *exec* command.

4. Correct answer and explanation: **A**. Answer **A** is correct, as you can download a large number of drivers from the CUPS Web site and easily check if the printer has a driver written for it.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect, as "Plug and Play" printers refer to their compatibility with Microsoft Windows and not to Linux. Answer **C** is incorrect; as there is no "CUPS Compatible" logo defined. Answer **D** is incorrect, as a lot of HP printers have Linux drivers, but not all do.

5. Correct answer and explanation: **A**. Answer **A** is correct, as this will allow you to print your job without affecting anyone else.

   Incorrect answers and explanations: **B**, **C**, and **D**. Answer **B** is incorrect, as there is no command to move your job to the top of the queue within CUPS. Answer **C** is incorrect, as like **B**, there is no pause command. Answer **D** is incorrect as you cannot force a job to the top of the queue using lpr.

CHAPTER 8

Installing, Configuring as a Server

Exam objectives in this chapter

- Network Services
- Web Services
- Application Services

INTRODUCTION

Servers provide network services to clients, such as Domain name server (DNS) and Dynamic Host Configuration Protocol (DHCP), or they serve as an application server, such as printing and mail. The main network and application services are explained in this chapter.

NETWORK SERVICES

There are many network services that can be deployed to a system, and this section will look at the most commonly deployed ones: DHCP, DNS, Network Time Protocol (NTP), and the interoperability with Microsoft Windows systems.

DHCP

DHCP servers are used to configure Transmission Control Protocol/Internet Protocol (TCP/IP) parameters for a host as they connect to a network, including

- IP address
- Name servers
- Configuring the routing, including the default route

Fast Facts

The main aspects of the DHCP you need to remember are as follows:

- DHCP allocates IP addresses to hosts on a permanent or temporary basis.
- Temporary IP addresses are said to be leased to a client by the server.

- Leased IP addresses can be renewed or relinquished.
- The list of leases is kept in *dhcpd.leases*, usually in /var/db.
- DHCP maximizes the usage of IP addresses available by allocating addresses on an as-needed basis.
- DHCP servers usually have fixed or static IP address.

DHCP SERVER CONFIGURATION

Before starting to configure the DHCP server, it is very important to understand your network and what parameters you will be configuring.

DID YOU KNOW?

The basic parameters you will want to configure with DHCP are as follows:

- Domain name
- DNSes
- Lease times
- Routing
- Static IP addresses
- Logging
- Primary or secondary DHCP server

The configuration file for DHCP is /etc/dhcpd.conf. Each subnet that you are going to provide DHCP services for must be defined in the file. The main options that can be used are described below:

```
# Sample configuration file for dhcpd
# Set the time a client can keep the IP address
default-lease-time 600;
max-lease-time 7200;
# set the default gateway to be used by clients
option routers 10.254.239.1;
# Set-up the NTP server
option ntp-server 10.254.239.6;
# set the nameserver to be used by the clients
option domain-name-servers 10.254.239.5
# This is a very basic subnet declaration.
```

```
subnet 10.254.239.0 netmask 255.255.255.224 {
range 10.254.239.10 10.254.239.20;
option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;
}
# Fixed IP addresses can also be specified for hosts.
# Names or IP addresses can be used
host adminprinter{
hardware ethernet 08:00:07:26:c0:a5;
fixed-address adminprinter.fugue.com;
}
```

This setup is allocating a number of IP addresses using DHCP, both leased and fixed.

DNS

The DNS resolves machine names to an IP address or it converts from the IP address to the name. In principle, a DNS resolves a name to its IP address.

Fast Facts

The Domain namespace can be regarded as structured in a tree form.

- Each domain within the tree is a node, with each node having a set of resource records (RRs) associated with it.
- RRs will define (at least) the ownership, name, and IP address.
- Domains can have subdomains, often referred to as children.
- The root is named . (dot), a corollary of the root drive / in the filesystem.
- Subdomains prepend their name to the root name, separated with another ".".
- Actual machines in each domain are defined by their machine name and the domain they are in or their fully qualified domain name (FQDN).

The Domain name system is a distributed database, and the nodes of this database are name servers with one or more authoritative DNSes that publish all the information about a domain, included in zone files on the DNS. The forward and reverse zones are defined as follows: reverse zones associate an IP address with a hostname and forward zones associate a name with an IP address. The DNS resolver is the client part of the client-server architecture and is the process that performs the resolution of the query, for example, the translation of the FQDN to its actual IP address.

DNS RRs

RRs are the most basic part of the Domain name system and have a number of basic elements to them: type, Time to Live (TTL), class, and possibly some data specific to the type of record. These records were described in Chapter 4, "Configuring the Base System," in the section "DNS Record Type and DNS Resolution."

CACHING NAMESERVER

A caching nameserver will build a local cache of resolved domain names and use this list to serve other hosts on your network. A large number of DNS requests are the same, and this will increase the speed of resolution and decrease the amount of traffic you send upward to another nameserver provided by *named*. A sample listing showing how a primary or master nameserver is set up is shown below, which is in the *named.conf* file (normally in /etc or /etc/named.d).

```
zone "mydomain.com" IN {

type master;

file "mydomain.com.xone";

allow-update [ none; ];

};
```

There is a control utility called *rndc* that allows you to administer the *named* daemon. The configuration file for *rndc* is /etc/rndc.conf, and additionally, you need to specify authentication keys in both /etc/rndc.conf and /etc/named.conf, which must match. You need to generate HMAC-MD5 keys for both the configuration files using the following command:

```
dnssec-keygen -a hmac-md5 -b <bit-length> -n HOST <key-file-
   name>
```

The default port used by *rndc* to connect to is 953. Once the command is set up, the *rndc* command can be used with the following options shown in Table 8.1.

NTP

The NTP synchronizes computer clocks over a network, including the Internet. There are primary (Stratum 1) and secondary (Stratum 2) servers with servers at a lower level that synchronize to a server at a higher level.

Table 8.1 *rndc* Options

Refresh	Refreshes the database
Reload	Reloads the zone files but keeps the cached files
Stop	Stops the service gracefully

Table 8.2 *date* Options

-a	Adjusts the date when the time is drifted
-u	Displays or sets the time in GMT
-s datetime	Sets the time and date

An NTP client is implemented as a continuously running daemon process, which runs in the kernel space due to the sensitivity of timing. The main configuration file is /etc/ntp.conf and defines the servers to synchronize to, as well as what networks are allowed to synchronize to, your server. A number of servers need to be defined to ensure redundancy, as shown below:

```
server  lerc-dns.lerc.nasa.gov  # Stratem 1 server
server  ntp.time.edu            # Stratem 2 server
```

The utility program *ntpq* is used to monitor the NTP daemon to determine the performance, and *ntpq –p* is useful to see what time servers are currently being polled. The local date and time can be set using *ntpdate* or *ntpd*.

DATE

date displays or sets the date on a system that does not have an NTP with common options shown in Table 8.2.

Windows Interoperability

Interoperability between Linux and Windows is normally an essential task that any system administrator needs to undertake.

REMOTE DESKTOP

rdesktop uses the Remote Desktop Protocol (RDP) and can be used to present remote desktops. To connect to a remote host, hostname.mycorp.com with IP address 10.10.100.23, either of the following commands can be used:

```
rdesktop hostname.mycorp.com
redesktop 10.10.100.23
```

Crunch Time

The target server or client must have the remote desktop connection enabled for this to work. In addition, you may need to supply username and password credentials applicable to the target host. The protocol runs on port 3389, which will need to be open on any intermediate firewalls.

VIRTUAL NETWORK COMPUTING

Virtual network computing (VNC) is a client-server application used to administer remote machines, operating on port 5901. A client for X Windows is *vncviewer*, which will connect to any VNC-compatible server.

SAMBA

Samba will implement the basic Server Message Block/Common Internet File System (SMB/CIFS) services, namely:

- File and print services
- Authentication and authorization
- Name resolution
- Browsing or service announcement

Users may wish to share some of all their files and allow only certain users to access (authentication and authorization). All these are handled by the daemon that is included within Samba.

Fast Facts

The two daemons included with Samba are *smbd* and *nmbd*.

- *smbd* handles file and printer sharing, including user access (authentication and authorization).
- *nmbd* undertakes name resolution on a point-to-point basis or broadcast basis, using the NetBIOS protocol.
- In broadcast mode, an *nmbd* client will send out a request to all machines on the network, for example, asking who is running a particular service.
- The other name resolution element of the *nmbd* daemon revolves around the NetBIOS Name Service (NBNS) or Windows Internet Name Service (WINS).
- Within NBNS, there is a master server that holds the IP address and NetBIOS name of each client or server on the network and will serve these upon request.
- The network browsing or service announcement part of Samba is also handled by the *nmbd* daemon.
- There is one local master browser (LMB) on a network that holds the list of available services and provide these upon request. These lists can be populated across domains via domain master browsers (DMBs).

Configuration Files

The main configuration file for Samba is smb.conf, usually residing in /etc/samba/smb.conf or /usr/local/samba/lib/smb.conf. The smb.conf layout is

similar to that used in older Microsoft Windows .ini files, comprising a number of sections with a section name in brackets ([]) delimitating the sections. The sections or stanzas will contain information about the shares, printers, and services on the server. There is one special stanza called *global*, which specifies parameters that apply to all other stanzas in the smb.conf file. A very minimal smb.conf file can be defined that just defines a couple of global parameters and some shares.

```
[global]

workgroup = mycorp

netbios name = computer_name

[share1]

path = /etc

comment = share the /etc folder to the world

[share2]

path = /documents

comment = share the global documents folder to the world
```

If you are setting up a server and want to share everyone's home directories, there is a special stanza called *homes*, which will enable the default home directory shares.

```
[homes]

comment = Home Directories

browseable = yes

Comment = only allow users to connect to their own directory, \\
   server\username

valid users = %S

comment = allow user to write to the directory

writable = yes
```

lmhosts File

The lmhosts file is built into Samba and is the NetBIOS name to IP address mapping, in a similar format to the /etc/hosts file. The file is located in the /etc/samba or /usr/local/samba/lib directories.

Managing a Samba Server

The Samba server has a number of daemons (notably, *nmbd* and *smbd*) that need to be started, normally upon boot, and will read the smb.conf file. Once started, the server can be managed from the command line or through a graphical user

Table 8.3	*smbstatus* Options
-b	Displays the list of users who are currently connected to the Samba server
-s	Displays the list of connected shares
-L	Displays the files that are currently locked
-u username	Displays information on the user *username*
-p	Displays a list of the *smbd* processes

interface (GUI). The command-line interface is very easy to use, and the main command is *smbstatus*, which can display the full status of the servers and connected clients. Some of the options available are shown in Table 8.3.

Connecting to a Samba Server

Assume that there is a Samba server located on the server *syngress* with a shared directory called *rosie*; you could map a drive on Windows to this from the command line using: *net use h: \\syngress\rosie.*

Linux has a client to access a Samba server called *smbclient*, with a syntax: *smbclient //servername/sharename.*

This client will display a new prompt to the user (typically, *smb:* \>) and will have very similar functionality to a File Transfer Protocol (FTP) session with *get, put, ls,* and so forth.

winbind

The integration of Linux and Microsoft Windows can be time consuming as there is no real underlying unified login. The *winbind* component of Samba tries to solve this by allowing Windows domain users to appear and operate as a Linux user. The mappings between the Linux and Microsoft Windows user IDs are stored by *winbind*.

WEB SERVICES

The following sections will show how Web services are configured and how they can be accessed by the client.

Fast Facts

The setting up of Web services on a Linux server includes FTP servers, proxy servers, and add-ons such as Java servers.

- Web servers will serve pages to the requestor using the Hypertext Transfer Protocol (HTTP).

- An FTP server will transfer data between a remote client and the server using the FTP.
- A proxy server will act as an intermediate server between a client and other networks, typically the Internet, to reduce the load on the connection.

Remote Access from the Command Line

The common utilities that can be used are *telnet*, *curl*, and *wget*. *Telnet* is a client-server protocol operating on port 23, and the syntax is *telnet hostname | IP address*, which will connect to a Telnet server at the *hostname* or *IP address*.

curl and *wget* retrieve files using HTTP, HTTPS, and FTP. The syntax for *wget* to download a URL is *wget URL* and for FTP is *wget ftp://URL*.

For sites that have FTP usernames and passwords, these can be specified in the command line, such as *–ftp-user=user*. As the command is not interactive, it can be built into scripts to automate the process.

curl can transfer data using a wide range of protocols and was designed to work without user interaction to facilitate its use in scripts. The syntax is *curl URL*.

Apache (HTTP) and Tomcat

The default installation of Apache will be /usr/local/apache2 and will be about 50 MB. The following sections will look at how you can configure the server.

APACHE CONFIGURATION

The actual directory where the server is installed is configured during the *configure* task and is changed from the default using *–prefix=PREFIX*, where *PREFIX* will be defined as the installation directory. This directory is referred to as the *ServerRoot*.

apachectl

The Apache server runs as the *httpd* daemon, and the control script *apachectl* should be used to invoke *httpd* ensuring that the correct environment variables are set. The syntax is *apachectl [http-argument]* with the main arguments being *start*, *stop*, *restart*, and *status*.

httpd

Upon starting, *httpd* reads its configuration file httpd.conf, stored relative to the ServerRoot in conf/httpd.conf. Once started, it will create a pool of processes to handle all the requests that are generated at the server. The main options are shown in Table 8.4.

Apache Modules

The Apache server can be customized with modules that can add a variety of functionalities to the Apache server such as bandwidth management, Common

Table 8.4 *httpd* Arguments	
-d ServerRoot	Sets the value for ServerRoot if different from the default
-k start | stop | restart	Starts or stops the daemon
-v	Displays the current version
-X	Runs in debug mode

Gateway Interface (CGI), or authentication. The modules that are currently loaded can be listed using

```
apache2 -l (shows those modules compiled in code)

apache2 -t -D DUMP_MODULES (show all loaded modules)
```

Modules can be enabled and disabled using the *a2enmod* and *a2dismod* commands.

Apache Containers

Within the Apache configuration files, there are individual units that contain directives that alter the configuration called *Apache containers*. The *filesystem container* contains all the directives regarding the directories and files, for example, location and access rights. The *Webspace container* contains all the information about the Web site you are developing, for example, the URL name.

As an example, to enable directory indexes for the /var/web/dir directory, the following would need to be included:

```
<Directory /var/web/dir>

Options +Indexes

</Directory>
```

The virtual host's container can be used when you have multiple hosts being served from the same machine, allowing different configuration options for each virtual host. The virtual hosts can be IP-based (one IP per Web site) or name-based (multiple names on a single IP address).

EXAM WARNING

You should know that the Apache Web files are located in the directory specified by the DocumentRoot directive specified in the httpd.conf file.

.htaccess Files

The .htaccess file allows the Apache server to have a decentralized management of its Web tree and has directives in the plain text configuration file. These directives will apply to the directory where the .htaccess file resides and is read

upon every access; so any changes will have an immediate effect. If the .htaccess file has options in it, the Apache configuration file must be configured with the *AllowOverride Options* set.

Hypertext Preprocessor

Hypertext **P**reprocessor (PHP) is a general purpose scripting language, mainly used for dynamic Web creation. When PHP starts, it reads its configuration file php.ini, usually in /usr/local/lib/php.

CGI Scripts

Apache can be configured to treat any file in a particular directory as a CGI script, referred to as the *cgi-bin* directory. The directory where the CGI scripts are held can be set up in the httpd.conf file using *ScriptAlias*. The syntax of this is

```
ScriptAlias URL Path Name
```

It is possible to run CGI scripts from any directory. The *cgi*-script handler must be activated using the AddHandler directive, and the ExecCGI directive must be specified in the Options directive. The httpd.conf file will need to be modified by adding the lines

```
AddHandler cgi-script .cgi
Options +ExecCGI
```

Configuring Apache Server Logs

The Apache error log is set by the *ErrorLog* directive, and it is the log file where the httpd daemon will send errors and diagnostic information to – normally called *error_log*. The LogLevel directive in the configuration file will define the amount of error logs or how verbose they are. There are eight levels of logs: emergency, alert, critical, error, warning, notice, info, and debug.

CRUNCH TIME

Do not allow anyone to write to the Apache log directory as this will almost certainly give them access to the UID that the server is started as, which is often root. In addition, the raw logs can have control characters inserted into it by malicious users, so care must be taken when viewing the raw files.

TOMCAT CONFIGURATION

Tomcat is a servlet and JavaServer Pages (JSP) container. JSP allows developers to create dynamically generated Web pages using HTML and XML. Servlets are the Java equivalent to PHP, CGI, and ASP.NET and can be automatically generated by a JSP compiler.

FTP

The FTP server is used to upload and download files to a server from an FTP client. An FTP server that allows anyone to connect to it is called an *anonymous server*.

Fast Facts

FTP servers can be set up in *active mode* or *passive mode*.

- In active mode, clients connect from a random port P > greater than 1023 to the port 21 on the server.
- Clients listen on port P+1 and send this information to the server using *PORT P+1*.
- The server will then connect to the P+1 port from its data port, port 20.
- Ports P and P+1 will be open on the client, and ports 20 and 21 will be open on the server.
- In passive mode, the client initiates both connections, opening ports P and P+1 (P > 1023).
- Port P connects to port 21 and then uses *PASV* to get the server to open a port above 1023, say S, and transmits this to the client using *PORT S*.
- The client connects to this port S.

An FTP server can restrict the local usernames that can be used, stopping users using, for example, root. This is achieved by putting the list of users in /etc/ftpusers, and this list of names must also appear in /etc/passwd. /etc/ftpchroot is in a similar format to /etc/ftpusers and contains the list of users whose session root directory needs to be changed, often listed in /etc/ftpd.conf, to ensure that the user cannot traverse into an unauthorized area.

Squid

Squid can be configured to cache HTTP and FTP traffic to reduce the amount of traffic destined for the Internet. Web clients are configured to access the Internet through this cache.

Fast Facts

Squid is a proxy server and Web caching service. It has many uses, namely:

- Caching Web lookups for the network
- Additional security for Web access
- Speeding up a Web server through caching of repeated requests.

SQUID CONFIGURATION FILES

The configuration file is /etc/squid/squid.conf, and the software is installed in /usr/local/squid. Squid can be configured to listen on any port, which is often useful if you are trying to hide some servers from general browsing. The default port for squid is 3128, although a lot of people change this to port 80. To allow squid to listen on ports 3128, 80, and 8080, change the squid.conf file:

```
http_port 3128 80 8080
```

Access control lists (ACLs) can be used which can restrict the networks that are allowed to connect to the server and with which protocol. The squid proxy can filter individual sites so that users cannot access them.

EXAM WARNING

Squid can be used as a proxy server and as a caching server. When it is a caching server, it displays data to the client that it has already in its cache, for example, a Web page. When it is used as a proxy server, it acts as a go-between for requests from a client to the target server. The target server will, therefore, not be able to communicate directly with the client.

APPLICATION SERVICES

The following section will explain the common application services of printing and e-mail.

Printing

The print server that is common within Linux is CUPS, and it is accessed via a Web browser using port 631, http://localhost:631, substituting the *localhost* part with a machine name or IP address if you are accessing a remote CUPS instance. Some pages require the root username and password.

NETWORK PRINTERS

Network printers require the IP address or can be found using Simple Network Management Protocol (SNMP) built into CUPS. There are three network protocols supported by CUPS:

- AppSocket Protocol, usually associated with HP JetDirect
- Internet Printing Protocol (IPP), normally port 631
- Line Printer Daemon (LPD) Protocol, port 515

Managing Operation Policies

There are rules built into CUPS to allow the administrator to define a number of policies, such as the user must supply a password, stored in *cupsd.conf* and are changed via the CUPS interface using the *Edit Configuration File* on the Administration tab.

Printer Classes

Administrators can group printers together to form a class, allowing users to send a document to this group, and CUPS will decide the best printer to use.

Mail

An e-mail server can have a number of different clients seamlessly connecting to it. The mail server for a domain needs DNS configured to ensure that there is a valid MX record.

The main transport mechanism between e-mail servers is the Simple Mail Transfer Protocol (SMTP), which operates on port 25. Mail is transferred from one e-mail server to another using a mail transfer agent (MTA), such as sendmail, or Post Office Protocol (POP) and Internet Message Access Protocol (IMAP) are used by e-mail clients to retrieve e-mail from a server.

HOW MAIL WORKS

Users will have usernames for a domain such as *username@your_domain*, and mail will be sent to the mail server at this domain as specified by the DNS MX record. When users send e-mail destined for a local user, the mail server will simply put the e-mail in the appropriate user's mailbox. If it is for someone outside the domain, the mail server will look up the MX record of the target domain and then try to send or relay the mail to that server.

Sendmail

The following sections will outline how to set up a sendmail server.

STARTING AND STOPPING SENDMAIL

Sendmail is usually started upon boot, when it reads the configuration file. Changes to the configuration file require it to be restarted.

```
service sendmail start
service sendmail stop
service sendmail restart
```

SENDMAIL CONFIGURATION

The main configuration file is sendmail.cf, normally located in /etc/mail/sendmail.cf or /etc/sendmail.cf. A common method of producing the sendmail.cf file is through the use of *m4* macro processor, which works on the configuration parameters in the file /etc/mail/sendmail.mc.

MAIL RELAYING

Mail relaying needs to be set up to ensure that your server is not used by spammers. You want to relay (or deliver) all mail that originates from your domain to the target domain. If your domain is mycorp.com, then you need to add this into the /etc/mail/relay-domains file. As it is relatively easy to spoof the

from address, /etc/mail/access can be used, which is more specific on who can use the relay server. This file can be used to configure a number of actions such as *relay*, *reject*, and *discard*. The file is a simple two column list, which needs to be converted into a sendmail compatible file.

POSTFIX

An alternative to sendmail is postfix, which was designed to be simpler to configure than sendmail. Although there are several hundred configuration parameters that are controlled by the configuration file /etc/postfix/main.cf., the variables are defined and used in a similar way to shell variables.

```
parameter = value

new_parameter =$parameter
```

The domain that is used in outbound mail is defined in the *myorigin* parameter, which defaults to the local machine name. The server can accept mail for a number of domains, and these will be specified in the *mydestination* parameter. As with sendmail, you must ensure you only relay mail from hosts or network you know and trust. This is undertaken using the *mynetworks* parameter.

The postfix daemon reports all errors to the syslog daemon, which itself sorts out events by class and severity. The logging classes, levels, and logfile names must be entered into /etc/syslog.conf to ensure these are logged correctly.

Fast Facts

You need to have a number of valid e-mail addresses "aliased" to another account.

- Aliases work for both sendmail and postfix.
- The aliases file is normally /etc/aliases or /etc/mail/aliases.
- Aliases are often configured to root.
- The aliases file needs converting to a format that can be parsed by the mail server, using *newaliases*.
- Aliases can be used to set up simple mailing lists.

MySQL

MySQL is a relational database management system (RDBMS) and will store data in tables, which can be linked together to enable manipulation of the data much easier.

MySQL CONFIGURATION

The main configuration file is /etc/my.cnf, and databases are located in a subdirectory of /var/lib/mysql. The MySQL data directory does need to be

Table 8.5 *mysql* Command-Line Options	
-u username	Connects a username to the database
-p	Prompts for password
-h hostname	Connects to the MySQL server on the remote host hostname

owned by the user which runs MySQL, and this directory should be set to 700 using *chown*.

STARTING AND STOPPING MySQL

The MySQL service can be started and stopped using the commands listed below:

```
service mysqld start

service mysqld stop

service mysqld restart
```

TESTING THE CONNECTION

The MySQL server can be tested very easily using the in-built command-line interpreter. The basic command is *mysql*, and the options are shown in Table 8.5.

mysql will attempt to connect to a remote database server using port 3306.

SUMMARY OF EXAM OBJECTIVES

In this chapter, you learned about how to configure a Linux system when it used as a server. The DHCP service provides the client with an IP address and other data, such as the local nameserver. The basics of DNS configuration were discussed, and how to set up the different files for forward and reverse name resolution was outlined. The interoperability with Microsoft Windows using a Samba server and how to modify the various configuration files to undertake this task were defined.

The section on Web services was centered around the Apache Web server, showing how the server is configured. The location of the main configuration files, the definitions of modules and containers pertinent to the Apache configuration, and how PHP and CGI scripts are incorporated into the Apache server were explained. The Squid proxy server configuration was shown, and how it is used in a network was described. The configuration of an FTP server and how this could be configured for different file transfers, such as straight ASCI text or programs in a binary form, were described.

Finally, mail servers and how to configure the two most popular servers, sendmail and postfix, were discussed.

TOP FIVE TOUGHEST QUESTIONS

1. You have been told that your mail MTA is being used as a relay by spammers. You want to stop this happening. What is the best course of action?
   - **A.** Relocate the mailserver behind your corporate firewall and only allow TCP port 25 to and from this server.
   - **B.** Ensure that the only hosts that the mailserver will allow to relay are on your local network by configuring the /etc/mail/access file.
   - **C.** Ensure that only your domain can be relayed by configuring the relay-domains file.
   - **D.** Configure the mailserver to stop all relaying of mail and make sure all the users connect to it via an approved client.

2. Your new Apache Web server has been set up and one of the developers wants to know which directory to load the Web pages. What directive will need to be accessed in the httpd.conf file?
   - **A.** WebRoot directive
   - **B.** DocumentRoot directive
   - **C.** ServerName directive
   - **D.** WebBase directive

3. You want to set up your Apache server to capture logs as you are having problem with the application. What log level would you set to give you the most verbose logs?
   - **A.** emerg
   - **B.** error
   - **C.** info
   - **D.** alert

4. The Samba server in your office has been set up with the name *samserv*. You want to connect to the sammy directory that has been set up and shared on it. What will be the correct command from a terminal shell if you want to connect as a user called juliet?
   - **A.** *smbclient //samserv/sammy juliet*
   - **B.** *smbclient //samserv/sammy –u juliet*
   - **C.** *smbclient //samserv/sammy –U juliet*
   - **D.** *smbclient //samserv/sammy U juliet*

5. You want to administer your DNS using the *rndc* command, but you cannot connect to the server. You have pinged the server and it responds. You have just installed the client on your machine. What is the likely error?
   - **A.** You have not put your machines IP address into the rndc.conf file for the target DNS.
   - **B.** You have not inserted the correct keys into the rdnc.conf file.
   - **C.** You have run the *dnssec-keygen* command immediately before issuing the *rndc* command.
   - **D.** You must run the *dnssync* command on the new host and the target to ensure they can communicate with each other.

ANSWERS

1. Correct answer and explanation: **B**. Answer **B** is correct, as this will specify which of your networks can connect to this mailserver and have their mail relayed. If you configure the networks correctly, the spam mail will not be relayed.

   Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect, as relocating the mailserver behind a firewall will not help. The server will have to have port 25 open on it, and this is used by spammers to force the mailserver to relay messages. This option would be useful if the correct configuration files as specified in answer **B** were done. Answer **C** is incorrect, as this will allow the domain users to still send e-mail, but spammers can easily spoof the domain name and could continue to use this as a relay mailserver. Answer **D** is incorrect, because if you stop the mailserver from relaying all mail, users will not be able to send any mail.

2. Correct answer and explanation: **B**. Answer **B** is correct, as the Document Root parameter correctly defines the root directory where the Web pages are stored.

   Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect, as there is no WebRoot parameter. Answer **C** is incorrect, as the ServerName parameter defines the name of the Web site. Answer **D** is incorrect because, again, there is no WebBase directive.

3. Correct answer and explanation: **C**. Answer **C** is correct, as this is the lowest log level and will give the most verbose output from the options listed. Debug can also be used, which will give a more verbose output than info.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect, as this will only give out emergency log message such as *Child cannot open lock file. Exiting*. Answer **B** is incorrect, as this just gives logs of errors. While this may be enough for your debugging, it does not give the most verbose logs. Answer **D** is incorrect, as this will just give log messages for actions that must be taken.

4. Correct answer and explanation: **C**. Answer **C** is correct, as this will invoke the smclient and connect to the share sammy on the target server (samserv). The response from the Samba server should be *Password*: prompting the user to enter the password they have set up on that host.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect, as the option to specify the user (-U) is missing. Answer **B** is incorrect, as the option to specify a username is –U, not -u. Answer **D** is incorrect, as this does not specify the user option correctly.

5. Correct answer and explanation: **B**. The correct answer is **B,** because the *rdnc* command needs to have authentication keys that match those of the target server.

Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect, as you do not need to insert the IP address of the target DNS in that file. Answer **C** is incorrect, because although you need to use this command to get the correct authentication key, the value must be put into the rndc.cong file. Answer **D** is incorrect, as there is no *dnssync* command.

CHAPTER 9

Securing Linux

Exam objectives in this chapter

- Managing and Monitoring User and Group Accounts
- File Permissions and Ownership
- SELinux Basics
- Implementing Privilege Escalation
- Security Applications and Utilities
- Checksum and File Verification Utilities
- Implementing Remote Access
- Authentication Methods

INTRODUCTION

Linux is regarded as a secure operating system; however, even the most secure systems can have an occasional flaw or be misconfigured, and security is best applied in layers. In this chapter, we'll look at the tasks necessary to make sure your Linux systems live up to their secure reputation.

MANAGING AND MONITORING USER AND GROUP ACCOUNTS

It's critical to be able to control who gets access to what information to maintain security and may even be a legal requirement. Limiting user accounts protects information and also protects the system itself, from both malice and simple errors.

Tools

The following tools are used to create and manage user accounts.

USERADD

There are a number of steps involved in adding users, and these may include the following:

1. Define the new user in /etc/passwd, and create a new User Identification number (UID). The system uses UIDs to refer to the user internally.
2. Create a password for the user bin /etc/shadow file.
3. Define a new group in /etc/group and a new Group Identification number (GID).
4. Create a new home directory for the user, set the file permissions on it, and copy the default startup files to it.
5. Set up the users e-mail account.

Although it is entirely possible to do each of these steps by hand, the *useradd* tool automates them. The syntax is *useradd [options] username.*

Good options to know are

- *-b* or *--base-dir*: location of the home directory.
- *-m* or *--create-home*: create a home directory for the user, and copy the basic user settings files from /etc/skel.

A number of other parameters can be set in the /etc/login.defs and /etc/default/useradd files, including

- Range of UID and GID numbers
- Location of the users e-mail file
- Account and password time limits

USERDEL

userdel deletes a user, and the syntax is *userdel [options] username.*

The options are

- *-r* or *--remove* deletes the user's home directory and his or her e-mail spool.
- *--f* or *--force* deletes the user account even if they are currently logged in, deletes the home directory even if other users may be sharing it, and deletes a group that matches the username even if it is used by others on the system.

Note that neither of these will remove files outside of a user's home directory.

USERMOD

usermod changes (modifies) a user account, and syntax is *usermod [options] username.*

Options include

- *-l* or *--login* changes a username.
- *-d* or *--home* changes the users home directory; use *-m* to move their files to the new location.
- *-u* or *--uid* and *-g* or *--gid* changes the users UID and default groupname or number but doesn't actually change the ownership information of any existing files they may have.
- *-G* or *--group group1[,group2,...]* changes the groups that the user is a member of. The user is removed from any groups not listed unless *-a* or *--append* is also used.
- *-L* or *--lock* and *-U* or *--unlock* options lock and unlock the account.

Crunch Time

As well as the above commands, there are some that work on groups only:

- *groupadd* creates groups, using the next unused GID number; *-g* or *--gid* lets you pick a specific number.
- *groupdel* deletes groups, removing them from /etc/group and (if used) /etc/gshadow but won't let you remove a user's primary group. *groupdel* won't change the actual GID information on existing files, and if the GID gets reused, you may get unexpected file access and other security issues.
- *groupmod* changes a group's GID or groupname, with *-g* or *--gid GID* changing the GID and *-n* or *--newname newgroupname* to change the groupname.

PASSWD

Users can use *passwd* to change their own password, or system administrators can change passwords on their behalf or to reset a forgotten password by providing the username. *passwd* can also be used to lock and unlock accounts with *-l* and *-u*.

EXAM WARNING

passwd isn't actually listed on the CompTIA list of exam topics, but they do refer to "lock," so it is advisable to be familiar with both *usermod -L* and *passwd -l*.

WHO *AND* WHOAMI

who lists the usernames of people logged into your system and can show where remote users are logged in from and the number of users. *who* works by looking in the /var/run/utmp file, which keeps track of who is logged into the system. The *whoami* command simply prints your username.

Fast Facts

w (syntax: *w [options] user*) provides the information *who* supplies, plus:

- What device they logged into?
- Where they logged in from (console or Internet Protocol (IP) address if remote)?
- When they logged in?
- How long their session has been idle?
- How much processor time they've used?
- What program they are running?

LAST

last reviews the /var/log/wtmp file to show information about who has logged in (and out) since the file was created. The syntax is *last [options] [name] [tty].*

Options include

- *-t YYYYMMDDHHMMSS* to show who was logged in at a specific time.
- *name* gives the log-in information for a specific user account. The system logs in with a "pseudo user" account called *reboot* each time it gets rebooted, so *last reboot* shows a list of times the system has been rebooted since the creation of the /var/log/wtmp file.

Files

We've touched on a number of files used to maintain user information; now, we'll take a closer look at a couple of them.

/etc/skel

Each user account has a home directory where his or her account settings and preferences are held. A set of template files, in the /etc/skel (short for "skeleton") directory, are copied to a new user's home directory by *useradd*. The template files can be used to give users helpful defaults for their BASH shell, standardized options for a company preferred text editor, and other standardized settings. It can even be used to give company standard browser favorites or a company directory file.

/etc/passwd

The /etc/passwd file is a text file but can only be edited by a user with elevated privileges. Each user has its own line, with fields by colons. The fields, in order, are

- username
- password
- UID
- GID
- comment(s)
- user's home directory
- user's default shell: "/bin/false" here means that user doesn't get shell access

/etc/shadow

/etc/passwd is used by many common utilities to cross-reference username and UID/GID information. Having easy access to every user's password, even if encrypted, is a bad thing. To solve that problem, there is a mechanism called a *shadow password file*; /etc/shadow can only be read by system administrators and contains the actual passwords. It can also be read by processes and utilities that are *setuid* 0.

EXAM WARNING

If you hand-edit the /etc/passwd file, be careful not to leave the *password* field blank. This clears the password, so anyone can connect as that username without a password. This is considered a poor security practice, even if other steps have been taken to limit that account.

/etc/group

/etc/group contains a list of groups and related information, one group per line, with fields separated by colons:

- groupname
- password
- GID
- members: separated by commas

Groups use a similar mechanism as users to set a password, which can be used to delegate group management to member users. *gpasswd* manages the passwords, which can be securely stored in the /etc/gshadow file.

FILE PERMISSIONS AND OWNERSHIP

The basic tools to manipulate permissions and ownership are described below.

Tools

There are a number of commands used to manage permission bits and other file attributes.

CHMOD

chmod is used to change the permission bits with syntax: *chmod [options] MODE FILE.*

Two useful options:

- *-R* or *--recursive* makes changes in the entire subdirectory tree.
- *--reference = file1 file2* sets the permission bits on file2 to match file1.

There are also two very different ways to represent the mode: using numbers and using letters.

Numbers use a three digit base-8 (octal) number, with the first digit for the owner, second for the group, and third for everyone else (other), as shown in Table 9.1. To set a file so only owner and group can read and write to it, you would enter *chmod 660 file3*.

When using the numbers, *all* the bits get set at once, where with letters you can adjust individual parameters one at a time. The specific format has a lot of combinations of options but is basically:

```
chmod who what_changes filename
```

The *what_changes* option comprises two parts: the action you want to perform (add, delete, and so on) along with one or more permissions, as shown in Table 9.2.

To allow the user and group to be able to execute a file, you would use: *chmod ug+x file3*.

CHOWN

chown is used to change a file's owner and group with syntax: *chown [OPTION] [username][:[groupname]] filename.*

Table 9.1 File Permission Value and Text

Octal	Permissions
0	---
1	--x
2	-w-
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

Table 9.2 Description of Options for *chmod*

Who	Action	Permissions
u for user	+ to add a permission	r for read
g for group	- to remove a permission	w for write
o for everyone	= to set all the permission bits as shown	x for execute
a for all the above		

EXAM WARNING

The execute bit has to be set before a binary executable file can be run, but because a script has to be opened for the interpreter to see the commands inside, it needs both the read and execute bits set.

The most useful options are the same as for *chmod*:

- *-R* or *--recursive* makes changes in the subdirectory tree.
- *username:groupname* filename changes the file's owner to username and the group to groupname.
- *username:filename* changes the file's owner to username and change the group to the users primary group (note the colon after the username).
- *:groupname filename* changes the file's group to groupname.

You can use the numeric UID or GID in place of a username or groupname.

CHGRP

chgrp works the same as *chown :group* and includes the same useful options of -R and --reference. Syntax is *chgrp* [OPTION] groupname filename.

CHROOT

chroot is a bit different than the preceding commands, as it doesn't change any of a file's attributes; instead, it changes how much of the file structure a program is allowed to see. It does this by redefining the top of the directory tree to where you specify. The syntax is *chroot newroot [command]*. This is normally used as a function within a script, but if the command is left off, *chroot* will give you a BASH shell with the root you specified.

LSATTR

lsattr displays a set of *attributes* that may be set on files and directories. These attributes define a number of advanced options and features that the computer uses when accessing information.

DID YOU KNOW?

The attributes include for a file are varied, described below:

- *i* means a file *immutable*; only the root user or privileged kernel processes can make changes to it. This essentially locks a file that you don't want change.
- *a* makes a file append-only, so it can only be added to, handy for log files.
- *d* marks a file to be skipped by backups.
- *c* marks a file for compression at the kernel level.
- *s* marks a file so that it gets overwritten with zeros when deleted, to enhance security.
- *A* tells the system not to use atime to update the access time on a file.
- *S* tells the system to immediately write any changes to the file, instead of caching them.
- *D* does the same as *S* but for directories.
- *u* marks the file to allow it to be undeleted.
- *H* indicates that a file uses special block sizing to allow it to be larger than 2 TB.

The syntax for *lsattr* is *lsattr [OPTIONS] [filename].*

Options include

- *-d* lists directories but not their contents.
- *-R* recursively lists subdirectories.

CHATTR

chattr is used to change the attribute bits that were covered in the *lsattr* section, with a syntax of *chattr [OPTIONS] [mode] filename.*

mode is a +, -, or = to add, remove, or set exactly a list that consists of valid option letters described in the *lsattr* section, above.

EXAM WARNING

Not all the available options may be incorporated into a particular Linux kernel and may have unexpected results even if they are. Additional research should be done before testing any of these on a production system.

UMASK

umask sets the default file permissions that a file gets when it is first created and uses a list of octal values to indicate what rights to *remove*. A typical *umask* is 0022, with the two's meaning new files will have the write privilege removed for members of group and other. You can view your *umask* by simply typing **umask** or change it by using *umask newmask.*

You can also use *umask* with the same letter syntax as *chmod*, by using the *-S* parameter which tells the system which bits to set, as opposed to which bits *not* to set for the number representation. To make a change permanent, you can add the command to your shell startup script, so it gets run every time you start a shell.

Special Permissions

There are additional special permissions that are represented by a fourth octal number, put in front of the normal three. Octal value of 4 is the *setuid*, 2 is the *setgid*, and 1 sets the *sticky bit*. They are set using the same *chmod* command as the normal bits, either using a four-digital octal value or the same method using letters, but with the following additional letters:

- *X* sets the user or group ID
- *s* restricted deletion flag
- *t* sticky bit

Fast Facts

The *setuid* or *setgid* bit can be set on directories or executable files.

- On directories, all files within the directory have the UID or GID of the directory.
- Setting the bits allows for shared directories.
- On executable files, the program runs with the privileges of the program owner or group membership of the program.
- Using the *setuid* bit allows files to be run with administrator rights.

STICKY BIT

When the sticky bit is set on a directory, files within that directory can only be renamed or deleted by their owner, the directories owner, or the system administrator. Without the sticky bit, any user with write and execute privileges in the directory could delete files.

SELinux BASICS

Operating systems are a complex set of modules, which will inevitably contain some flaws that need to be patched at some time during their lifecycle. The real risk is that someone figures out how to exploit a flaw or vulnerability in your system before you get a chance to patch it or even before the software provider realizes there is a problem and can fix it.

SELinux reduces the risk of these problems by implementing a system that limits what programs can do. Ideally no user or program should have access to anything more than it needs to do its job – this is a standard security paradigm called *least privilege*. SELinux is not another Linux distribution, as it is built into the Kernel (from 2.6 onwards). It incorporates mandatory access controls (MAC), a mechanism that enforces least privilege access by program. Normally, a Linux system uses discretionary access controls (DAC) that depend on the user to set the security level.

Using DAC, a program run by a user will have access to everything that user has. Therefore, if the root account (or a process that runs with that privilege) is compromised, the entire system would be compromised. With MAC, the security policies are mandatory and set by the system owner. Even with elevated privileges, the security policy on files and programs cannot be overridden.

Fast Facts

To make it easier to set things up, SELinux provides three running modes:

- *enabled*: SELinux is up and running, any forbidden actions are blocked.
- *disabled*: SELinux is there but not turned on.
- *permissive*: SELinux is up and running, with the rules in place, but when something forbidden is attempted, SELinux allows it.
- Initially, SELinux sets to permissive mode to identify conflicts easily.

IMPLEMENTING PRIVILEGE ESCALATION

As a system administrator, you need to be careful what you do; even a small typo can cause devastation. Consider the difference between *rm -r /tmp/test* and *rm -r /tmp/test*. The difference of a single space causes the system to try and delete every file on your machine. To protect your system as well as to use the idea of "least privilege," it's advisable to use your system with normal user privileges whenever possible. When extra authority is called for, you can use *privilege escalation*.

Fast Facts

There are two ways to escalate your privileges: *su* and *sudo*.

- *sudo* is put in front of commands, and after you press the **Enter** key, it will ask for your regular user password and run the command as administrator. Once you authenticate, *sudo* will remember the password for a few minutes, so additional uses don't ask for a password.

- Only users who are listed in the/etc/sudoers file can use privilege escalation. It is recommended that you use *visudo* as root to make any changes, which opens the sudoers file in *vi* and protects it from simultaneous edits.
- It is often easier to start an entire new command shell with administrative privileges with the *su* command. *su* uses the administrator password.

SECURITY APPLICATIONS AND UTILITIES

The following tools are open source and available online and are all very complex and feature rich. For the exam, it is important to know what each tool is used for.

EXAM WARNING

These tools are very useful for protecting your systems, but like any power tool, they can just as easily be misused. Be sure you have full authorization – preferably in writing – before using them outside of your own test environment. The tools that send out test packets are capable of sending information that can cripple or reboot some systems, which is a great way to test if a system is vulnerable, but could lead to a lot of problems. Please use these tools with care.

Fast Facts

There are numerous tools to test for vulnerabilities on a system and to ascertain what ports are being listened to. The main ones are

- *nmap* is a network scanning tool and is invaluable for seeing what is attached to your network. It can test for services or open network ports, finding unauthorized network devices and services and testing firewalls.
- *nessus* tests networked systems for security vulnerabilities and works by using a list of currently known security problems, which is kept updated by means of downloaded plug-ins.

wireshark

wireshark is a graphical network traffic analyzer built on the text-based *tcpdump* utility and monitors all network traffic coming in and out of an interface on your computer. It is very useful for fixing network applications, such as client/server database problems, remote access authentication issues, and printing errors. Frequently, the client software hides error details from the user, and a lower level view is needed to isolate the real issue.

snort

snort is a Network Intrusion Detection System (NIDS) that watches traffic and lets you set a list of traffic you might find interesting, or you can download lists of known signatures of traffic. If an interesting packet wanders by, *snort* can send out an alert, so you can then investigate further.

Tripwire

Tripwire is a Host-based Network Intrusion System, and when first installed, it takes a digital "thumbprint" of key files (more detail in the next section), and occasionally, it checks to make sure that they haven't been tampered with.

CHECKSUM AND FILE VERIFICATION UTILITIES

One complex security problem involves being able to trust that files you rely on haven't been tampered with. Every Transmission Control Protocol (TCP) packet passing through every router is checked; the router "adds up" the bits, and if they match the total that the sending device claims it did, everything is assumed to be okay. When it comes to security, though, it requires protection against possible malicious intent not just random errors.

Fast Facts

A number of methods are currently used to certify that files haven't been tampered.

- *md5sum* uses the MD5 algorithm to calculate a checksum for a downloaded file, which is compared to a checksum supplied by the file's creator. If they match, you have a good copy.
- *sha1sum* has similar functionality to *md5sum* but uses the SHA-1 algorithm.
- *md5sum* uses 128 bit encryption, and *sha1sum* uses 160 bit encryption.

gpg

gpg (Gnu Privacy Guard) uses an open implementation of pretty good privacy (PGP) to encrypt and/or sign files using public/private key pairs. Although it is normally used for signing and encrypting e-mail to make sure it hasn't been read or changed, it can also be used to sign files. It works like this:

- The sender of a file creates a public/private key pair, keeping the private key secret and sharing the public key with anyone who will need to decrypt or

verify information from him. The key generation process only has to be done once. Sharing the key can be done in-person or using a key-escrow service.
- The recipient of the file obtains the senders public key and imports it into his system.
- When the recipient gets a file, he or she uses the sender's stored public key to decrypt or just authenticate the file.

IMPLEMENTING REMOTE ACCESS

Secure methods of remote access are defined below.

SSH

Although *Telnet* is still available, the preferred method for accessing a command shell on a remote system is with the secure shell (SSH). An *ssh* client is used to connect to a remote system running the *sshd* daemon. The syntax for ssh is *ssh [OPTIONS] [username@]hostname [command]*.

Basic connectivity to connect to server1 is to type **ssh server1** and assumes you want to use the same username on the remote system, as you are using on the local one. To log in to the remote system with a different username, you can either use the *-l username* option or put the username in front of the host name, with an @ between them, like this: *ssh bob@server1*

The first time *ssh* is used to connect to a host, the new hosts signature is shown, and *ssh* asks if you'd like to add the signature to your list of known hosts.

SECURE TUNNELS

ssh can catch traffic going to a local TCP port, pass it through its own encrypted connection, and hand it off to a TCP port on the remote side, encrypted normally unencrypted network traffic. This is called *forwarding* or *tunneling*.

To create a tunnel, you need to know the TCP port of the service on the remote server that you want to tunnel to and pick a random unused TCP port on your local service. The syntax looks like this:

```
ssh -Llocal_tcp_port:localhost:remote_tcp_port remote_host_name
```

SFTP

ssh can be used instead of File Transfer Protocol (FTP), which doesn't use any encryption, using *sftp*. Once you log in to the remote system, you can then use *sftp* commands to send and receive files. The syntax to get connected in this way is the same as regular *ssh*.

Fast Facts

SFTP commands are similar to both BASH and FTP, including

- *cd path* changes the remote directory.
- *ls* views the contents of the remote directory.
- *put* copies a file from the local to the remote machine.
- *get* copies a file from the remote machine to the local.
- *bye* or *quit* exits *sftp*.

X11 FORWARDING

One feature of *ssh* is X11 forwarding, which is essentially the same as the TCP port forwarding discussed earlier. It allows you to connect your local X11 server to the remote X11 client such that a program running on the far end draws a graphic interface on your local screen. To test it, log into a remote host with *ssh* or *ssh –X* if X11 forwarding is not enabled, and execute a program that has a graphic interface. You may want to add an *&* at the end of the command to run it in the background.

KEYGEN

The *ssh-keygen* command is used to create a public/private key pair. Once the public key is placed on the remote host, *ssh* uses the keys to authenticate your log-in, and passwords are no longer required. The steps are

1. Use *ssh-keygen* to create a key pair.
2. Copy the public key from your local user home directory .ssh/id_rsa.pub to the remote user home directory .ssh/authorized_keys.
3. *ssh* checks for matching keys when logging in, and if they are found, it doesn't ask for a password.

Once the keys are in place, *ssh* can be configured to require public/private keys to log in remotely, which makes for a very secure system, as long as your keys remain safe and don't get lost.

AUTHENTICATION METHODS

Authentication is the process the system uses to determine you are supposed to be given access when you type in your username and password. It comes in two basic flavors:

- Local authentication is limited to a single computer. It knows who you are, but no other computers do. It is easy to set up and administer on a few computers but scales poorly.

- Centralized authentication allows user information and other settings to be gathered into a single repository and then accessed from trusted computers. Centralized systems can be much more complicated to configure, but make it much easier to administer large networks of computers.

PAM

Pluggable Authentication Modules (PAMs) provide dynamic authentication requests from Linux programs and services. The modular design means that implementing some new authentication technology (like a fingerprint scanner) or policy (like mandatory password complexity) is as easy as plugging in the appropriate modules and telling the system to use them by updating the appropriate configuration file. Each authentication program has its own PAM configuration file. Configuration files are stored in the /etc/pam.d directory.

Fast Facts

There are four groups of management tasks that are covered by PAM:

- Authentication modules verifies users.
- Account modules checks that the account is still valid, tracking items such as passwords, account expiration, and time of day.
- Session modules handles tasks related to the start and close of a user session. Sessions cannot start until the user is authenticated.
- Password modules are used to update password or other authentication mechanism and can enforce the strength of the password.

Each PAM configuration file lists types of management task, which module(s) to check while performing the task, and a control that tells if passing the module test is, among other checks, mandatory or optional. The actual PAM modules are stored in /lib/security/.

LDAP

Lightweight Directory Access Protocol (LDAP) is used to query and modify directory services on TCP/IP. A directory can be considered to be a set of objects, each with their own attributes which are organized in a hierarchical manner. The LDAP directory tree often uses DNS to structure the upper levels of this hierarchy with specific organizational structures below this which could be organizations, teams, individual people, or even hardware such as printers.

LDAP systems typically use the stand-alone LDAP daemon; *slapd* provides the back-end server functionality to store user information. Individual workstations then access the *slapd* directory information as needed: usually via a PAM plug-in for authentication or maybe through an e-mail client to look up e-mail addresses.

NIS

Network Information System (NIS) is another option for storing centralized user and other configuration files. It was originally called the **yellow pages**, so many *NIS* commands and files start with "yp." NIS has a central directory of information containing users, groups, host names, and e-mail addresses as well a plethora of other information. There is a database of configuration files that can be shared across the network, and as /etc/passwd, /etc/group and /etc/shadow are shared, and a user can log-in on different workstations.

RADIUS

Remote Authentication Dial In User Service (RADIUS) was originally developed for dial-in access via modems but is now used as centralized authentication and authorization system across networks. It is a client/server protocol with the client forwarding requests to the RADIUS server to grant or deny the request.

Two-factor Authentication

Two-factor authentication refers to a requirement to provide two things when you log in, usually "something you know" – a password – and "something you have" – an access token of some sort. Two-factor authentication makes it much harder to break into someone else's account because just guessing his or her password is no longer good enough.

SUMMARY OF EXAM OBJECTIVES

Although "security" is frequently associated with "restrictions," it's helpful to consider it in terms of "allowances." To summarize, remember that the *adduser* command creates accounts to allow access to the system, *chmod* modifies what users can do with files, and *groupadd* can be used to create groups that can be used to allow users to share information. The *su* and *sudo* commands extend user privileges to allow administrative tasks, and *ssh* allows users access to other systems. User account information can be shared between systems with *ldap* or *nis*, and authentication requirements can be customized using on a Linux system with *pam*.

There are number of applications for observing the Linux system environment, including

- *Nessus* and *nmap* for scanning networks
- *Wireshark* for capturing packets on a network interface
- *Snort* for watching a network for suspicious traffic
- *Tripwire* to watch for suspicious file updates

SELinux is used to limit what an application can do using MACs, and *gpg* can be used to encrypt information or to guarantee it hasn't been altered.

TOP FIVE TOUGHEST QUESTIONS

1. You need to create a new group to support a new product roll-out. What command, or commands, will let you make a new account?
   A. *addgroup project_x*
   B. *groupadd -g project_x*
   C. *newgroup project_x*
   D. *groupadd project_x*

2. You want to tighten security on a particular Linux computer by limiting which users have access to the *sudo* command. Which file should you edit to lock down this feature?
   A. /etc/users
   B. /etc/shadow
   C. /etc/sudoers
   D. /etc/passwd

3. You need to set a shared file for read and write access for the file owner and members of the file group and no access for anyone else. Which command(s) will give the desired result?
   A. *chmod 440 shared_file*
   B. *chmod 660 shared_file*
   C. *chmod ug=rw,o= shared_file*
   D. *chmod og=r,e= shared_file*

4. You are testing out SELinux to enhance security on your Linux computer. What mode would you use to let all programs run, but log anything that would fail if you were to lock it down?
   A. enabled
   B. allowed
   C. permissive
   D. test

5. You are running out of room on your backup system and want to flag a large temporary file so the tape backup system skips it. What is a way you could do that?
   A. chmod -s temp_file
   B. setattr -d temp_file
   C. chattr -d temp_file
   D. attr -d temp_file

ANSWERS

1. Correct answers and explanations: **A** and **D**. Answer **A** is correct because the *addgroup* command is used to create a new groups on some distributions. Answer **D** is correct because *groupadd* is used to create a new group on other distributions. It is more generally supported than *addgroup* but doesn't have as many options.

Incorrect answers and explanations: **B** and **C**. Answer **B** is incorrect because the *-g* option is used to specify a numeric GID not an alphanumeric name. Answer **C** is incorrect because *newgroup* isn't a valid Linux command.

2. Correct answer and explanation: **C**. Answer **C** is correct because *sudo* security is controlled by the /etc/sudoers file.

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect because /etc/users isn't a standard Linux file. Answer **B** is incorrect because /etc/shadow is used to store encrypted user passwords. Answer **D** is incorrect because /etc/passwd is used to store user account information but not *sudo* access rights.

3. Correct answers and explanations: **B** and **C**. Answer **B** is correct because the octal mode bits 660 equate to read (4) and write (2), totaling 6 for both user (first position) and group (second position), with no rights (0) for everyone else (third position). Answer **C** is correct because it sets rights for user (u) and group (g) to read (r) and write (w) and sets everyone else (o) to nothing by leaving the field blank. Note that the different groups of users need to be separated by a comma.

   Incorrect answers and explanations: **A** and **D**. Answer **A** is incorrect because octal mode 440 would set owner and group rights to read only. Answer **D** is incorrect because the file owner is correctly abbreviated with a "u" (think user) and everyone else is represented by an "o" for "other," not an "e."

4. Correct answer and explanation: **C**. Answer **C** is the correct because in permissive mode, SELinux is engaged but doesn't block programs, only logs what it would block if it were set to "enabled."

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect because in enabled mode, SELinux prevents programs that violate defined policies from running. Answers **B** and **D** are incorrect because allowed and test modes aren't valid SELinux running modes.

5. Correct answer and explanation: **C**. Answer **C** is correct because the *chattr* command is used to set file attributes in Linux, and the *-d* option is used to flag a file to be skipped by backup systems (*d* references *dump*, a basic backup program).

   Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect because *chmod* is used to change file permissions not attributes. Answers **B** and **D** are incorrect because *setattr* and *attr* are not valid Linux commands.

CHAPTER 10

Troubleshooting and Maintaining Linux

Exam objectives in this chapter

- Monitoring Tools
- Analyzing Logs
- Backing up and Restoring

INTRODUCTION

This chapter covers common tools for managing a system, including monitoring performance and logs and backing up data. If your system seems to be running slowly or having some other issues, it's very helpful to have some historical data to compare, so it is advisable to run a few of them now and then under normal load, so you have a good idea what "normal" looks like.

MONITORING TOOLS

Linux provides a number of handy tools for reviewing system status and other statistics. Not all of them may be included in a default installation, but they are all supported and can be made to work with a little effort.

Commands

The usage of the most popular utilities is described below.

SAR

sar can be used to gather both current and historic system statistics for the system processor. By default, *sar* displays information periodically gathered and stored in files at /var/log/sysstat/sa*, with the last two numbers of the sa* files corresponding to the calendar day the information was collected. Some other useful options include *sar -A* to show all statistics and *sar -n DEV* to show network statistics. An example showing network statistics is shown in Figure 10.1.

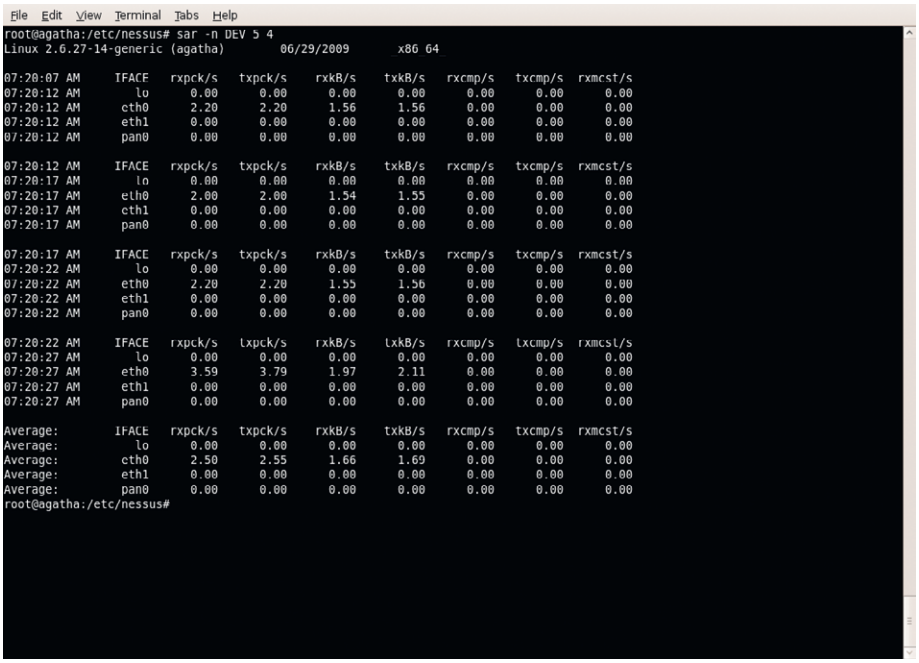

FIGURE 10.1
An Example of Using *sar* to View Network Statistics

Fast Facts

You can control the collection of current information by appending two numbers at the end of the command.

- The first will set the collection interval in seconds.
- The second tells how many times it should gather the information.
- *sar -n DEV* 5 4 shows network statistics for 5-s time periods a total of four times.

IOSTAT

The *iostat* command shows both CPU and disk utilization statistics and can also show remote Network File System (NFS) drive systems. It was covered in Chapter 5, "Using BASH."

VMSTAT

The *vmstat* command uses information in /proc/meminfo, /proc/stat, and /proc/*/stat to show virtual memory and other system usage statistics, including disk and processor usage. *vmstat* can be followed with two numbers indicating how long to wait and how many times to run. Memory statistics using units of

blocks equate to 1024 bytes of memory per block in current Linux kernels. Using *vmstat* without options gives a brief overview of memory and CPU statistics. The *-d* option shows more detailed statistics related to the disk drives in the system.

UPTIME

The *uptime* command gives a quick one-line display showing the current system time, how long the system has been up, the current number of users, and the load average over the last 1, 5, and 15 min. The load average is a count of processes either currently being handled by the processor or waiting to be run.

TOP

The *top* command shows a current running tally of system usage and an ongoing list of processes. This is covered in Chapter 5, "Using BASH."

Load Average

Many of the commands that show system utilization display three numbers that represent the system *load average*. You can see it in the upper section of the *top* command and on the end of the *uptime* command, and it also shows up in the *w* command.

> **EXAM WARNING**
>
> You may have noticed in the exam objectives that the focus is on commands; however, there are several key statistics that are objectives as well, load average being one. You should be familiar with the commands to know which ones will display load average and other utilization statistics.

The load average is a three part statistic that indicates, on average, how many processes are currently running or actively waiting to be run by the processor and is essentially a count of how many programs are waiting to be run at a given moment, with a damping factor to give a more accurate time-weighted average. The 1-min average has a smaller damping factor and is allowed to swing more quickly, whereas the 15-min average has a higher damping factor and gives a better long-term overview of the system load.

ANALYZING LOGS

A record of events is kept in a variety of log files, and these files can be crucial for troubleshooting problems, tracking down failures, and finding security issues. Different distributions may have slightly different log files and directory structures. The *syslog* configuration is defined in the /etc/syslog.conf file and is used to determine which system processes record their events to what files and what level of logging to use.

Table 10.1 Syslog Event Levels

Level	Description
Emerg	Panic situations
Alert	Urgent situations
Crit	Critical conditions
Err	Other error conditions
Warning	Warning messages
Notice	Thing that might merit investigation
Info	Informational messages
Debug	For debugging only

System events are categorized into eight levels, listed in Table 10.1, and can be produced by one of 21 predefined facilities or system programs.

Crunch Time

The following log files and their usage are very important for the exam.

- /var/log/messages: It is the standard location for most system events. All info, notice, and warning messages are sent here by default, along with some authorization and cron events.
- /var/log/syslog: It collects information about authorization events, time changes, and system statistics events if you installed the *sysstat* package.
- /var/log/maillog: It is the events related to the mail service.
- /var/log/secure: It is used by Red Hat-based distributions to record authorization messages for *sshd*, *sudo*, and other authorization events.
- /var/log/lastlog: It is the information about user login times and is a binary file. *lastlog* uses this file to show the last time a user has logged in to the system.

Rotating Logs

Over time, log files keep growing and would eventually fill whatever hard drive they are written to, so it is important to implement a log management scheme. The standard program for rotating log files is *logrotate*, and by default, it is run daily by the *cron* schedule. It can be set to rotate, compress, remove, and/or mail log files based on specified times. The /etc/logrotate.conf file contains the options used by *logrotate*.

Searching and Interpreting Log Files

Most log files are simple text with valuable data that can be read with your favorite paging application, such as *less*. If you are searching for a specific type

of event or troubleshooting a particular issue, all the other events will simply be cluttered and get in the way.

Fast Facts

There are a number of useful commands to search log files.

- The output of a command can be piped into *grep* and used to filter what shows up on the screen, or you can feed files directly into *grep*. The syntax is *grep [options] PATTERN [file]*. When used, *grep* will search the named *file* (or standard input) for lines that match the *PATTERN* and prints any matching lines. The number of options available is large, and *regular expressions* can be used.
- *tail* shows the last 10 lines of text in a file, or with the *-n NUM* option, it shows the last NUM lines. The *-f* option *f*, for follow, refreshes the screen with new information as the log file gets updated. It may be convenient to open a separate terminal window just for running *tail -f*.
- The *awk* utility is a programming language specialized for text and string functions. The general syntax is *awk [options] -f script-file filename.* If the filename is left off, *awk* will use STDIN. *awk* is much more flexible than *grep*.
- *sed* is a *stream editor*, intended to edit files using scripts, and is useful for doing bulk search and replace functions within multiple files and acts on files one line at a time. The syntax is *sed [options] -f script_file filename.*

BACKING UP AND RESTORING

Backups can be done at different levels. File level backups run at the operating system level and are convenient for dealing with individual files, but they require a working operating system to restore to. Some software offers online and offline backups. An online backup uses the database or e-mail system to read each record or mailbox and make a copy. An offline backup requires the software that manages the database or the e-mail system be shut down. It is also possible to do an offline backup of entire drive systems and computers themselves by booting them from an alternate device.

Fast Facts

There are two general kinds of backups – complete backups and partial backups.

- Complete backups back up all the data you have selected.
- Partial backups back up only the data that has changed since the most recent complete backup.

- Partial backups are divided into differential and incremental.
- Differential backups contain everything that has changed since the last complete backup.
- Incremental backups copy everything that has changed since the last partial backup.
- To restore data from incremental backups, you need the last complete backup and all the incremental backups.

Copying Data

File-level copies of data over a network can be done using *rsync* and FTP. By copying over a network, it is possible to easily send your backups to off-site locations, although bandwidth may become an issue.

RSYNC

rsync doesn't just copy data, but it synchronizes the data, so if you use *rsync* on a directory, the other end will be made to be an exact duplicate of the original. It can read data in blocks and only copy the portion of files that have changed, called the *delta-transfer*, which means it won't copy an entire file if only a small part of it has been modified. *rsync* can either be run using a remote shell or be configured to run as a service, using its own Transmission Control Protocol (TCP) socket and transport system.

The basic syntax for moving files using *rsync* is shown in Table 10.2.

FTP

Another common way to transfer files is by using *FTP* and can be used for backups. Using *FTP* requires both a client and a server that the client connects to and was discussed in more depth in Chapter 8, "Installing, Configuring as a Server."

Table 10.2 *rsync* Command Syntax

File Operation	*rsync* Command Syntax
Copy files locally from one directory to another	*rsync [OPTION...] SRC... [DEST]*
Pull files via remote shell	*rsync [OPTION...] [USER@]HOST:SRC... [DEST]*
Push files via remote shell	*rsync [OPTION...] SRC... [USER@]HOST:DEST*
Pull files via *rsync* daemon	*rsync [OPTION...] [USER@]HOST::SRC... [DEST]*
Push files via *rsync* daemon	*rsync [OPTION...] SRC... [USER@]HOST::DEST*

Archiving and Restoring Commands

The following commands are the standard utilities for backing up data; each having their own strengths and weaknesses.

CPIO

The *cpio* program uses binary archive files. It has three basic modes:

- copy-out mode creates an archive.
- copy-in mode reads from an archive.
- copy-pass mode transfers files from on place to another without creating the actually archive as a middle step.

In copy-out mode, *cpio* accepts a list of files that you want to put into your archive from STDIN and sends the output to STDOUT, so putting *cpio* to use requires some command-line redirection. The output can be a regular file, device file, or network location.

In copy-in mode, *cpio* reads an archive file from STDIN and spits out the files into the current directory. The specific files to be extracted from the archive can be selected using a pattern, where *pattern* is standard filename wildcard. The following commands demonstrate how this is done using *cat* and *cpio*:

```
cat archive_file | cpio -i "pattern"
```

TAR

tar is used to store files in and extract files from an archive file, called a *tarfile*. This tarfile may be created and stored on any rewriteable medium, such as a tape drive or hard disk. The following is the syntax for creating a tarfile with *tar*:

```
tar c [ bBeEfFhiklnopPqvwX [ 0-7 ] ] [ block ] [ tarfile ]
  [ exclude-file ] {-I include-file | -C directory | file | file }
```

The most commonly used options for *tar* are listed in Table 10.3.

Table 10.3 Commonly Used Options for *tar*

Option	Description
-A, --catenate, --concatenate	Append tarfiles to an archive
-c, --create	Create a new archive
-d, --diff, --compare	Find differences between archive and filesystem
--delete	Delete from the archive
-r, --append	Append files to the end of an archive
-t, --list	List the contents of an archive
-u, --update	Only append files that are newer than copy in archive
-x, --extract, --get	Extract files from an archive

DID YOU KNOW?

The *tar* command has a long history and is very versatile, being used for creating archives that are used for software source packages and being used as a way to back up and extract files.

- *tar* was originally designed to make tape archives.
- *tar* can be used to archive files to tapes or disks.
- tarfiles can be compressed.

DUMP

dump can archive entire filesystems and works by examining files on a target filesystem and determining the files that need to be backed up. These files are then copied to the backup medium, usually hard disk or tape, and if the *dump* is larger than the backup medium, the *dump* is split and copied to multiple volumes. On most media, the size is determined by writing until an end-of-media indication is returned.

The following is the basic syntax for *dump*:

```
dump -0 -A [archive file] -f [destination file or device] [mountpoint
   of a filesystem, or a list of files and directories to be
   backed up]
```

In the example above, the *-0* option tells *dump* to perform a full backup. The *-A* option is used to designate the archive file, which is read by the *restore* command (described below) when restoring the backed up files. *-f* is used to identify the target file or device that will host the "dumped" files.

RESTORE

restore performs a restore of files that have been backed up using *dump* and can be used to restore a full backup of a filesystem and apply any subsequent incremental backups. *restore* can be used to restore individual files or directories from a full or partial backup file archive and can operate across a network to restore filesystems or files and directories on remote systems.

The following code is the basic syntax for *restore* that will restore all files from the identified archive (mounted at /dev/nst0) in the current directory.

```
restore rf /dev/nst0
```

In this example, the *r* option retrieves all files from the archive and the *f* option is used to designate the archive (/dev/nst0).

DD

dd is useful for making and copying disk images, backing up and moving disks, and duplicating filesystems, as it makes an exact clone of a hard disk, including all blank space. To use it, the source disk must be unmounted and the output destination must be at least as large as the source.

To illustrate, the following syntax will create an image file in my home directory entitled, cdbackup.iso, from the CD in my CD-ROM drive:

```
dd if=/dev/cdrom of=/home/brian/cdbackup.iso
```

The option *if* designates the input file (the CD that is mounted in my CD-ROM drive), and *of* designates the output file, cdbackup.iso in my home directory.

Writing to Removable Media (CD-RW, DVD-RW)

Creating CDs and DVDs is a relatively commonplace activity that people do for a variety of reasons: backing up files on a computer, creating a photo CD, and transferring music and videos, among others. At a high level, there are two steps in the process of burning a CD or DVD:

1. Create an image file for the CD or DVD, which involves creating filesystem on the medium and adding data to an image file.
2. Apply the image to the CD or DVD.

Regardless of the media, nothing can be stored until a filesystem has been created on it. Unlike a hard disk, a CD-R or DVD-R is writeable only once, which means that if you create the empty filesystem as a step by itself, it will remain empty forever. Burning a CD or DVD involves creating a filesystem while transferring the files to the medium. The command that accomplishes the first part of the process – creating an image file (an .iso file) that includes both the filesystem and the actually files that will be transferred to the medium – is *mkisofs*. You can use the following syntax for creating a typical CD or DVD image:

```
mkisofs -r -o [filename of CD or DVD image.iso] [directory where files
   to be copied are located]
```

The option *-r* sets the permissions of all files on the CD or DVD to be public readable and enables RockRidge-extensions. If you do not use the *-r* option, the files will be copied with the permissions that were assigned to them in their original locations. The second step is to apply the image to the CD or DVD using a separate program, *cdrecord*. The following syntax will mount your image file at the mount point, /cdrom:

```
mount -t iso9660 -o ro, loop=/dev/loop0 [filename of CD or DVD image]
   /cdrom
```

The *-t* option, iso9660, identifies the filesystem of the image file as that of a CD or DVD. Once the CD or DVD image file is mounted, you can navigate to the /cdrom directory (using *cd /cdrom*) and verify that you have included all the files you wanted and that the directory structure is sound. Then, insert the appropriate blank media in your CD or DVD burner. You will need to figure out the SCSI device address of your CD or DVD burner. As root, issue the command *cdrecord –scanbus,* and then you can issue the command (substituting the correct device) address if necessary:

```
cdrecord dev=0,0,0 [filename of CD or DVD image.iso]
```

SUMMARY OF EXAM OBJECTIVES

There are so many monitoring and measuring utilities available on Linux that there is no excuse not to monitor your system. The *sar* and *top* programs will produce detailed reports on how different components of your system are performing at specified intervals and in real time, respectively. You will want to pay special attention to the programs that calculate and display the load average: *top*, *w,* and *uptime*, among others.

Although monitoring will help you check up on how things are going with your system, you will need to know where to go to find information when the inevitable happens and things go badly. There are more logs; many of which reside under the /var/log directory structure, and knowing the various logs under /var/log is important. With the amount of data that is captured in each log, you need good searching and sifting tools such as *grep*, *sed*, *awk*, and *tail*.

Restoring data that has been lost due to a hard disk crash or user error is inevitable. Creating backup sets with *tar* and *dump*, *restoring* files with *tar* and *restore*, and using the *dd* command to create disk images and apply them to other media were covered.

TOP FIVE TOUGHEST QUESTIONS

1. You have been asked to back up users' data on a particular server before a core application is upgraded. Because of the amount of data, you need ensure that these files will fit on a remote hard disk. What command would you use to ensure that the smallest possible size of the backup file?
   A. *tar -cvf userdata.tar /home/**
   B. *tar -xjvf userdata.tar /home/**
   C. *tar -cjvf userdata.tar /home/**
   D. *tar -xvf userdata.tar /home/**

2. You are replacing Michael's computer and have backed up the hard disk to an external hard disk (/mount/usbhdd) using the following syntax: *dump -0uf -A michaelhdd.archive -f /mount/usbhdd/michaelhdd.backup /*. You want to restore the backup on another hard disk in the new computer. After booting the new computer and mounting the external hard disk, what command do you use?
   A. *restore -rf /mount/usbhdd/michaelhdd.backup*
   B. *dump -xf /mount/usbhdd/michaelhdd.backup*
   C. *tar -xvf /mount/usbhdd/michaelhdd.backup*
   D. *restore -rf /mount/usbhdd/michaelhdd.archive*

3. Lately you have been hearing reports that your Linux server is slow to respond and you have a suspicion that there are applications that are consuming more than their fair share of the server's memory. What key

combination would you press while *top* is running so that the running programs are sorted by their respective percentage of memory utilization?
A. *F*, then *M*
B. *F*, then *n*
C. *F*, then *k*
D. *F*, then *l*

4. Users are reporting that a particular corporate server responds slowly for around 30 min between 10:30 A.M. and 11:00 A.M. You decide to run *sar* at regular intervals during this time to capture statistics on the server's network performance. What syntax would you use to capture six sets of these metrics every 10 min?
A. *sar -A DEV 600 6*
B. *sar -n DEV 600 6*
C. *sar -n DEV 6 600*
D. *sar -A DEV 6 600*

5. You are in the process of setting up an active mode FTP server. Whenever you try to connect, you can connect to the server, but you cannot enter a username and password. You made sure that TCP ports 21 and 20 are open on the server. What is the most probable cause of the problem?
A. TCP ports 1022 and below are open on the server
B. TCP ports 1023 and above are open on the server
C. TCP ports 1022 and below are closed on the server
D. TCP ports 1023 and above are closed on the server

ANSWERS

1. Correct answer and explanation: **C**. Answer **C** is correct because the command with the *-c* and *-j* switches creates the archive and compresses the files in the archive with *bzip*, respectively.

Incorrect answers and explanations: **A**, **B**, and **D**. Answer **A** is incorrect because while the *-c* switch creates the tarfile, there is nothing to instruct *tar* to compress the files in the archive. Answer **B** is incorrect because the *-x* switch is used to extract the files from the archive, which in this case contains compressed files. Answer **D** is incorrect because the *-x* switch is used to extract the files from the archive.

2. Correct answer and explanation: **D**. Answer **D** is correct because the *restore* command with the *-rf* switches is required to extract all files from the archive and specify the archive file.

Incorrect answers and explanations: **A**, **B**, and **C**. Answer **A** is incorrect because the output file is specified and the *-f* switch is used to specify the archive file. Answer **B** is incorrect because *dump* is used to back up the files and has no restore functionality. Answer **C** is incorrect because only *restore* can be used to extract files from a backup set created with *dump*.

3. Correct answer and explanation: **B**. Answer **B** is correct because *n* will display the percentage of memory usage.

   Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect because *M* will display the CPU time in hundredths of a second. Answer **C** is incorrect because *k* will sort by percentage of CPU utilization. Answer **D** is incorrect because *l* displays utilization based on CPU time.

4. Correct answer and explanation: **B**. Answer **B** is correct because the switch and parameters needed to collect the desired statistics *-n DEV 600 6*. The *–n* switch tells *sar* to capture network statistics, the next parameter, 600, specifies the interval in seconds, and the last interval, 6, specifies that six sets of statistics should be captured.

   Incorrect answers and explanations: **A**, **C**, and **D**. Answer **A** is incorrect because *-A* tells *sar* to capture all available system statistics. Answer **C** is incorrect because the last two parameters are inverted, which means that 600 sets of statistics will be captured at 6-s intervals. Answer **D** is incorrect because the *-A* switch is used, and the parameters are inverted.

5. Correct answer and explanation: **D**. Answer **D** is correct because TCP ports 1023 and above need to be open to ensure that all the connections required for active mode can be established.

   Incorrect answers and explanations: **A**, **B,** and **C**. Answers **A** and **C** are incorrect because active mode FTP uses TCP ports 1023 and above, not below. Answer **B** is incorrect because the connection could be established if the ports were open.

Glossary

The following terms show up throughout the book and, with their accompanying definitions, they should be a useful resource when studying for the exam.

Apache containers Sections within the Apache Web server configuration files.

Apache modules Add-ons for the Apache Web server.

Attributes The properties of a file, such as size, modification date, ownership, and permissions.

Binary package Group of files in compiled format that has been designed to work on the appropriate Linux distribution.

Child process Process spawned from a parent process. Each parent can have many child processes.

Class (of printer) Group of printers linked together so they can act as one.

Command Line Interpreter or Command Line Interface (CLI) A full-screen or windowed text-mode session where the user executes programs by typing in commands with or without parameters.

Common UNIX Printing System (CUPS) The standards-based, open source printing system.

Dependencies Packages required to be installed when another package is installed.

Dynamic Host Configuration Protocol (DHCP) A protocol for automating the configuration of computers that use TCP/IP.

Discretionary Access Control (DAC) Type of access control where system privileges are specified by the owner of an object, who can apply, modify, or remove them at will.

DNS server, authoritative A DNS server that gives the definitive answer to a DNS query, and not a cached answer from another server.

Domain Master Browser (DMB) The domain master server within an individual samba workgroup.

Exit value Shows the exit status of the shell and commands.

Firewall Security device to segregate networks.

Forwarding or tunnelling One network protocol encapsulating a different protocol.

FTP server, active mode FTP client uses the PORT command.

FTP server, anonymous server An FTP server in which users do not need an account on the server.

FTP server, passive mode FTP client uses the PASV command.

Fully Qualified Domain Name (FQDN) Complete definition of a host, such as *host.test.com*.

Grand Unified Bootloader (GRUB) Bootloader used on most modern versions of the Linux operating system.

Group Identification number (GID) Identification number of a Linux group.

Hardware Abstraction Layer (HAL) Abstraction layer between hardware and the software in a computer.

Hostname Name of the specific host.

Immutable Files that cannot be deleted, even by root.

Internet Message Access Protocol (IMAP) An e-mail protocol that allows users to receive using an IMAP e-mail client, where all e-mail, folders, and so on are stored on the server, rather than on the client's local computer.

Inode A data structure holding information about files in a Unix file system.

Iostat Program to report statistics regarding the I/O system.

Journaling Journaling filesystems keep a record of changes and allows for faster restart after an unexpected system shutdown.

Kernel The core operational code of an operating system.

Lease Temporary assignment of an IP address to a host by a DHCP server.

Least privilege Runs programs with as few privileges as possible.

libraries Groups of files.

Linux distribution A particular version of Linux, such as OpenSUSE, Red Har, and so on.

Load average The average loading of the system over a period of time.

Local Master Browser (LMB) Master browser on a subnet.

Logical partition Subpartition of a primary partition.

Makefile Describes the dependencies between files.

Mandatory Access Control (MAC) A type of access control where system privileges are specified by the system.

Master Boot Record (MBR) Holds the partition table of the disk and can contain executable code.

Metadata Data about data.

Mode Permission on a file.

Mail Transport Agent (MTA) A program responsible for delivering e-mail messages.

Network Time Protocol (NTP) The TCP/IP protocol used to synchronize the clocks on computers across a network.

Package manager Collection of tools to automate the installing, upgrading, and configuring of software packages.

Packages Collection of files, either in binary or source form.

Parent process The process that spawned (started) a new process.

Post Office Protocol (POP) An application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server.

Privilege Escalation Process to escalate your privileges to that of an administrator.

Redirection Redirects IO from or to a different device.

Regular Expression (regex) A recognized method for describing a search pattern.

Relay Method of forwarding e-mail on.

Residual data Small parts of a disk that have obsolete data.

Root Often called superuser or administrator.

Runlevel A specialized script that starts a different set of services.

Samba An open source suite of programs that provides file and print services to SMB/CIFS clients and allows for interoperability between Linux/Unix servers and Windows-based clients.

Server Message Block (SMB) Protocol that can be used to interface linux and windows systems.

ServerRoot In Samba, location where other files can be made relative to.

Shadow password Encrypted password file.

Software repositories Locations where different software packages are held.

Source package Collection of files that need to be compiled.

Stanza Section of a Samba configuration file.

Sticky bit Programs run with the permissions of the owner of the file and not that of the user who runs the program.

Stream editor Noninteractive editor.

Syslog Utility for logging system messages.

Terminal A utility that can be run to input commands into.

User ID (UID) A numeric identifier that represents a user.

Window Manager A program or suite of software that controls the placement and appearance of windows within a windowing system in a GUI.

X Window System (or simply, X) An open source suite of software (including a network protocol) that implements the X display protocol, provides windowing and manages keyboard and mouse functions to provide a GUI for networked computers.

Index

T

U

V

W

X

Y